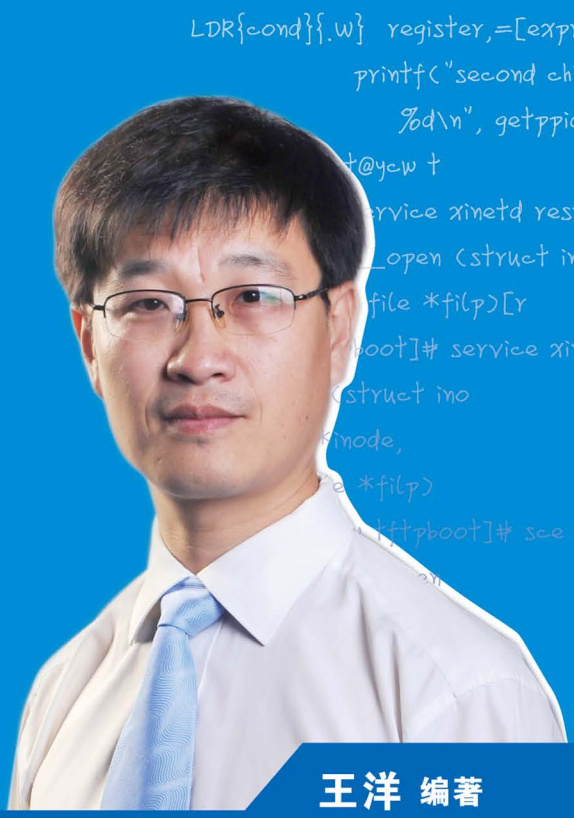
教育是用生命影响生命的过程

Broadview®
www.broadview.com.cn

Java Web开发

就该这样学

我用了**8年时间教授**Java Web这门课，尝试着让初中毕业生、被传统教育淘汰的人，或是退休在家的老人掌握编程技术，**大约15000名**各种基础和理解能力的学生轻松地掌握了这项技术，令人欣慰的是所有的学生从始至终都**将这个学习过程当成一场游戏**，并将编程这个工作发展成自己的兴趣。在这本书，我将阐述我与众不同的做法，希望学习者能够在一开始便建立起新的学习思想，**这样我们才能一起玩代码**。



王洋 编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

金牌讲师为您开启编程的快乐之旅

Java Web开发就该这样学

王洋 编著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书基于建构主义教育思想，通过大量循序渐进的案例，让学生在体验中掌握 Java Web 相关知识，同时获得编程能力、排错能力和学习能力，本书多次使用陷阱式教学法，帮助学生深刻理解所学知识，掌握实现 Web 编程的不同技术特点。

本书详细介绍了 Java Web 程序设计的前端技术、开发和部署，以及一些衍生技术变化。在内容上，本着使用不同技术尽可能实现相同功能的原则，让读者能够充分体会认识每个技术的优缺点。

本书的内容和组织形式立足于高校教学教材的要求，适用于从职业院校到重点本科院校的教师教学和学生学习，可以作为 Java Web 程序设计的入门教材，或者面向就业的实习实训教材，同时可作为计算机技术的培训教材，读者完全可以通过本书自学 Java Web 技术。

图书在版编目（CIP）数据

Java Web 开发就该这样学 / 王洋编著. —北京：电子工业出版社，2013.6
ISBN 978-7-121-20453-1

Ⅰ. ①J… Ⅱ. ①王… Ⅲ. ①JAVA 语言－程序设计 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2013）第 103759 号

策划编辑：孙学瑛
责任编辑：徐津平
印　　刷：三河市鑫金马印装有限公司
装　　订：三河市鑫金马印装有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×1092　1/16　印张：15.5　字数：365 千字
印　　次：2013 年 6 月第 1 次印刷
印　　数：4000 册　　定价：49.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

一直以来人们都认为教师和书籍是知识的载体，教学的过程就是将这些知识传递给学生，于是书上写满了正确的知识，学生看书就可以迅速掌握知识，理论上这是高效率的系统，但事实上只有极少数人能够适应这样的系统，因为这些知识也是有人经过一个过程得到的，忽略了发现知识的过程，而直接将结果传递给学生，似乎高效率，但是学生却常常无所适从，因为学习是发现知识的过程，而不是记住知识的过程。

好在建构主义教育思想指明了更加适合学生的教学过程，在这一思想下，书和教师从正确知识的传递者，转变成探索知识的引领者，带领着学生去体验、去感觉、去发现属于学生自己的知识，正如这本书，读者会发现有 60%的内容是不正确的，这些错误是学习过程避免不了的，优秀的学生不同之处就是，通过自己的努力在到达学习目标的道路上，不断地调整，将错误的理解剔除掉，问题是大多数学生无法完成这个过程，要么陷入错误的包围中，最终放弃了探索，要么通过死记硬背来自己达到学习目的，结果学生能够通过考核，却没有运用知识的能力。学习离正确的轨道越来越远，甚至很多人迷失了学习的真正目标，将记住知识作为唯一的目标，很多教育者苦苦探索的正确教育途径，在教育理论研究中早已经被发现，那就是建构主义教育。

建构主义教育思想从来不认为掌握知识是学习的最终目的，我认为学习的目标是认知、能力和精神。认知和知识是不同的，知识停留在人的头脑中，而认知是能够被熟练使用的知识；能力在不同的领域是不同的，由于 Java Web 技术是前端页面技术和后端编程技术的结合，其中前端技术结合了 HTML、CSS 和 JavaScript，这些几乎完全不同语法特点的编程，通常是 Java Web 程序员的薄弱环节，所以综合运用能力就成为 Java Web 前端技术学习的关键。

后端编程技术经过多钟不同思路的发展，产生了更加适合编程的 Servlet 技术和 JSP 技术，并且在两个主流的分支基础上又演变出了 JSP+JavaBean 以及 JSP+Tag 的应用手段，为了改善用户体验，还出现了 AJAX 技术，所以在这本书的范畴内，一个优秀的 Java Web 程序员，既需要有前端编程能力，又需要掌握后端编程的特点，清晰地理解每一种技术适用的范围和优缺点；如果学习的目的仅仅是为了掌握一项技术，那么人终将被新的技术手段所替代，任何学习过程都是生命价值的提升，一个程序员需要有严谨的态度、专注的品质、探索的精神和创新的意识。这些学习目标不是一节课或一个章节的任务，需要通过整个教学过程来建构。

一直以来学生的学习动机都是教育理论界热衷讨论的话题，我认为学生学习的动机有三个方面，一是为了获得喜悦，二是为了消除恐惧，三是自我效能。好的成绩可以获得家长、老师的表扬，可以有更好的名次甚至奖学金。而差的成绩会被批评、留级，甚至拿不到毕业证。我们发现普遍的教学手段是为了推动学习动机的前两个方面，这造成了两个可能的结果，有些学生对于奖励或是惩罚麻木了，一旦丧失了学习动机，自然好的成绩无从谈起，在另外的学生身上，这些手段一直能够起到作用，我们会得到所谓的好学生，问题是这些动机是外界推动的，而非内生的，这些习惯于此的好学生或许一生都在意别人的评价。如果教学过程能够激发学生的自我效能，让学生的学习是基于自己强烈的爱好和成功的喜悦，我们就一定能够培养出来优秀的学生，而他们也将一生受益。

问题是为什么建构主义教育思想如此的好，却很少在教学实践中应用，这是因为建构主义和现有的教学形式相比仍有些弱点：第一，建构主义在教育的效率理论上比较低，现在我们能够在短时间内将大量的正确知识传递给学生，学生只需要理解记忆就好了，而建构主义教育要呈现知识探索的过程，这样会消耗更多的时间和精力。第二，建构主义教育的效果不可控，学生是通过体验自己发现整合知识，那么不同的学生或许得到的结论不同，深度不同。第三，考核困难，我们不能再用知识点来考核学生，因为教学过程中就没有传递经典的知识点。第四，实施建构主义教育对于教师的要求比较高，教学过程的设计建立在对学生深入理解的基础上，教师不仅仅要准备教学知识了。

为了实现上述效果，老师将扮演不同以往的角色，教师不再是知识的载体，教师将陪伴着学生一同探索，带领着学生犯错误，引导着学生进行思考整合。为了克服建构主义教育思想的弱点，在写这本书时，我基于对学生和技术的理解，剔除了大量知识点的讲解，在反复的教学实践中，已经能够获得和传统教学相同的教学效率。另外我大量总结和研究了学生的学习过程，建立了学生在学习 Java 过程中的学习曲线，依照学习曲线来评估和考核学生的学习效果。

本书总结了作者多年在这条道路上的探索，力求提供基于建构主义教育思想的 Java Web 教学材料，帮助学生轻松地掌握作为 Java Web 程序员所需要的知识和能力，通过比对运用相似技术，帮助学生将最适合的技术运用到项目中。书中的内容并不是简单的案例堆砌，每个部分的任务都包含了对相关知识的整合，都基于学生的学习曲线特点。

我在 8 年教学探索后才动手写这本书，因为我一直相信“教育是用生命影响生命的过程”，我无法在一本书中实现和我亲自上课同样的影响过程，课堂上一遍遍的重复代码所传递的严谨态度，无法在书中呈现，加上我对技术、对学生理解，以及对于建构主义教育思想理解的局限，让我清楚地知道，我并没有完成一部让我心满意足的作品，书中不可避免的有很多不足，恳请读者批评指正。

这本书的内容是我数以万计的学生的成果，甚至有很多案例是我的学生在学习的过程中发明的，这段从 8 年前开始的探索并不是一蹴而就的，我诚挚地感谢我所教过的学生，是他们的忍受、包容和努力帮助我完成了这本书。我要感谢我的家人，我儿子的出生和成长，让我开始接触和研究教育理论，给我之前漫无目的地探索指明了方向，为了让这本书通俗易懂，我那学文科的

爱人像一名真正的学生一样，通过这本书来学习 Java 技术，在她的努力下，这本书具备了更强的覆盖范围，确保读者即便是没有任何专业基础，也能够通过这本书掌握 Java 技术。同时也要感谢电子工业出版社的老师们为本书的出版所付出的辛勤工作。

光盘使用：我一直希望这本书的定价尽可能低，希望有更多的人能够没有负担的学习 Java 技术，本意不想提供光盘，但是权衡再三，由于本书中的代码是伴随着讲解逐步展开的，很多代码没有整体呈现（否则会增加太多页数），所以不得不通过光盘提供书中的代码，这些代码被放在 codes 目录中，按照书中的章节组织，需要强调的是，请不要直接编译运行，或是复制我提供的代码，光盘中的代码是我的，只有你亲手输入到电脑里的内容才属于你。

王　洋

于 2013 年 4 月 14 日

CONTENTS

目　　录

第 1 章

认识 Tomcat

1.1 什么是 Web Server

今天来讨论 Web 对大多数人来说都不是多么陌生的概念，我们在电脑中打开浏览器，在地址栏中输入 www.sohu.com，就能够看到搜狐公司提供给我们的网页，来思考一下在你的操作背后，计算机本身以及 Internet 为我们在做些什么事情。

首先是浏览器，经过 Java 部分编程的学习，你不应该对计算机里面的任何程序有所敬畏，那些程序都是和你一样的程序员编写出来的，理论上只要你愿意，花些时间你也可以编写出相似的程序！

那么，如果由你来编写一个浏览器，你会做些什么？我们能够想象，用户输入的 www.sohu.com 是一个网址，它对应了一个 IP 地址，通过网址得到一个 IP 地址，并不是一件多么难的事情，互联网上的 DNS 服务就可以提供这样的转换。作为一个程序员，得到一个 IP 地址后，会创建 Socket 对象和对方连接上，目的是为了得到一个特定的网页，我们常常能够在浏览器的地址栏中看到这样的网址：http://www.360buy.com/product/258313.html，意思是我要看京东网站上的 258313.html 这个文件的内容。这是由于一个网站上的网页文件太多了，所以还会有路径结构。

www.sohu.com 这样的网址请求是个例外，你并没有指定要看哪个网页，通常网站会提供一个默认的网页，如果没有指定，就将这个默认的网页传送给你。无论是哪种形式，我们将浏览器中输入的地址称为 URL。

我们不看这样的例外情况，来想一下，如果要看 258313.html 这个网页，是不是要通过 Socket 将这个请求传送到京东的网站服务器去呢。

京东的网站服务器一定是一台计算机，但是只有计算机是不够的，计算机上需要运行一个程序，我们能够理解，这个程序将是服务器端的 Socket 应用，里面有 ServerSocket，这个程序将监听在一个端口，大家约定了提供 Web 服务的端口是 80，当然也可以改变这个约定，那么在浏览器访问的时候，就要额外指定新的端口号，如果不这样做，就意味着浏览器访问的是 80 端口。

服务器的程序将接收到浏览器发出的请求，然后它用 IO 流到自己所管理的目录中找到这个 258313.html 文件，将这个文件的内容读到内存中，然后通过网络发送给你的浏览器，浏览器

得到了这个文件的内容，将按照 html 的规则显示出来。这就是一次最简单的 Web 访问的过程，如图 1-1 所示。

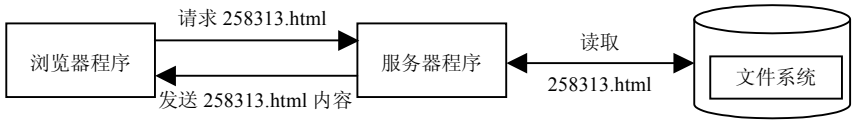


图 1-1

作为一个计算机的使用者，浏览器是一个非常常用的程序，有很多公司会把这样的程序做成产品，所以现在我们已经不需要来编写这样的程序了。

那么服务器程序在互联网世界中，是不是也是非常常见的程序呢，其实编写服务器程序更加有利可图，所以现在在市场上有很多服务器端的软件产品，这些产品被统称为 Web Server。

1.2 选择 Tomcat 来学习

比较著名的有这样几种，在微软的服务器端操作系统中，会附带提供名字叫做 Microsoft Internet Information Server（简称 IIS）的产品，由于微软公司的成功，这款产品被广为使用。但是它有两个弱点：一是，它通常只能运行在 Windows 系列产品中，虽然在个人计算机中 Windows 的普及率非常的高，但是作为服务器端的计算机，使用 Windows 系统的情况相对少得多；二是它不能直接支持 Java。

Tomcat 却没有这样的问题，Tomcat 能够运行在任何操作系统中，也是 Java 官方推荐的 Web Server，当然它不支持微软所倡导的 ASP 或是 ASP.NET。Tomcat 还有一个非常重要的优点，这个软件是免费的，这并不意味着 Tomcat 是简陋的，事实上有很多著名的网站是由 Tomcat 支持而建立起来的，基于这些优点 Tomcat 成为学习 Java Web 的首选 Web Server。

此外还有 BEA 公司的产品 WebLogic，IBM 公司的产品 WebSphere，这些都是大名鼎鼎的 Web Server，它们往往提供了一些 Tomcat 所不具备的功能。问题是它们太昂贵了，当然这些额外的功能并不是我们现在这个阶段所涉及的。

1.3 安装 Tomcat

Tomcat 是一款开源的免费软件，这意味着 Tomcat 的版本升级速度是非常快的，问题是每次 Tomcat 的升级，都会带来配置和使用细节的变化。所以你新下载的 Tomcat 和这本书中的一些细节，有可能不是那么吻合，但是这个现象并不打紧，因为基本的思想和方法并没有太大变化。

截止目前，Tomcat 的最新版本是 Tomcat 7，虽然对于初学 Tomcat 的人来说，这个版本的升级离我们还有些遥远，但是我确实想不出来拒绝一个最新版本的理由，下载 Tomcat 的官方网址

是 tomcat.apache.org，你可以找到在国内的下载地址，当然即便是在官网上下载也是非常快的，我们也发现这里出现了一个名字叫做 Apache，我不能称 Apache 是一个公司的名字，我们通常将其叫做 Apache 组织，Tomcat 是这个组织提供的一款产品，随着学习的深入，你未来可能还会遇到这个组织的其他产品。我下载的版本是 Tomcat 7.0.25，文件分为 32 位版本和 64 位版本，你需要根据自己的计算机来选择。

下载下来的文件是.zip 的压缩包，Tomcat 是一个纯绿色的软件，不需要安装，只要将这个压缩包解压就可以了。解压后的目录如图 1-2 所示。

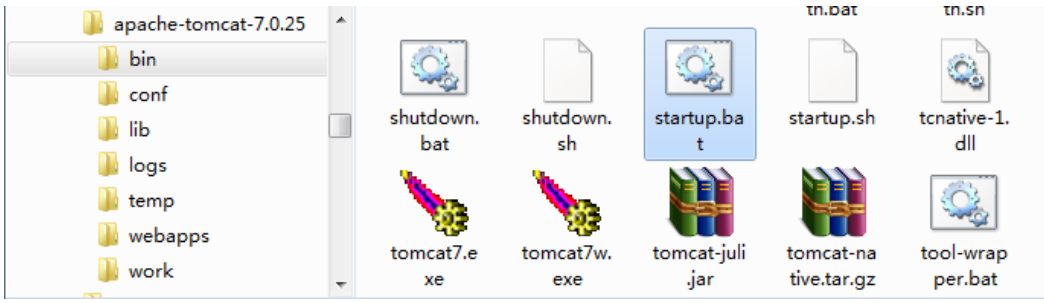


图 1-2

选择放在 Tomcat 目录下的 bin 目录，并选择其中的 startup.bat 文件，.bat 文件是微软系统环境下的批处理文件，通常人们将一系列的命令放到这个文件中，这样就能通过运行这个批处理文件，同时启动其中的命令了。这个文件用于启动 Tomcat 程序，像我们熟悉的那样，双击运行这个程序，你会发现有一个黑色的控制台窗体一闪而过，我抱歉的告诉你，这代表运行是失败的。

Tomcat 和 JDK 的联系的紧密程度超乎想象，如果没有 JDK 的支持 Tomcat 根本就无法运行，可是 Tomcat 自身并不带 JDK，它需要找到你计算机中的 JDK，由于我们没有安装，也就是说没有一个安装程序能够帮助 Tomcat 找到 JDK，所以我们要按照 Tomcat 的要求指定 JDK 的位置，Tomcat 需要一个叫做 JAVA_HOME 的环境变量。

设置环境变量的过程在上一本书《Java 就该这样学》的开始进行过类似的讲解，用鼠标右键单击“我的电脑”图标，选择“属性”命令，然后在“高级”选项卡中找到“环境变量”按钮，“系统属性”对话框如图 1-3 所示。

单击“环境变量”按钮，会弹出一个设置环境变量的窗体，如图 1-4 所示，窗体分成上下两个部分，上面那个部分设置的是影响当前用户的环境变量，而下面设置的是影响这台计算机上所有用户的环境变量，可根据你的情况来选择，如果这台计算机是你的，选择那个都无所谓。

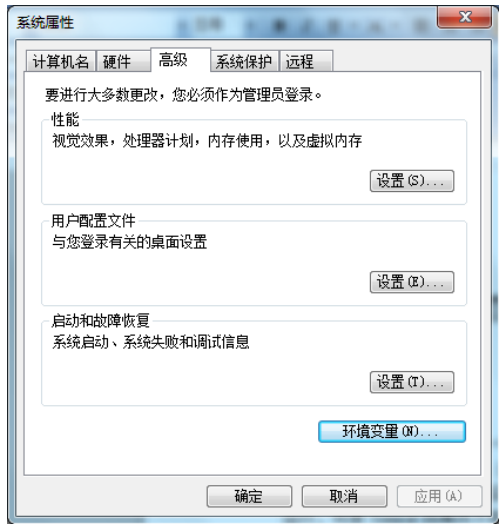


图 1-3

单击“新建”按钮，输入环境变量名为 JAVA_HOME，你可以通过资源管理器找到你正在使用的 JDK，然后将路径复制到变量值输入框中。

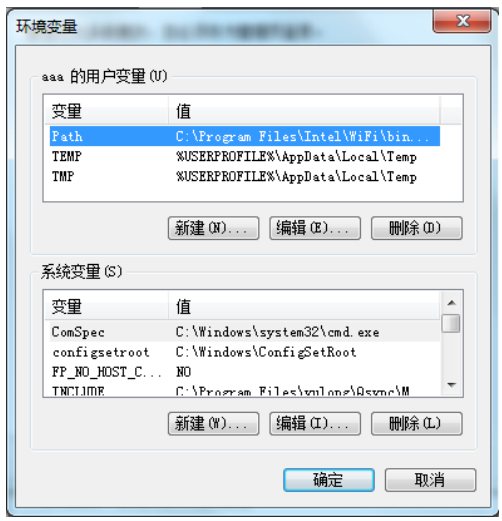


图 1-4

单击“确定”按钮后，再回到 Tomcat 目录中，到 bin 目录下找到 startup.bat 文件，双击运行它，经过一番等待后，你会看见在一个黑色的控制台窗体里面出现很多输出，如图 1-5 所示。

见到这句 Server startup in 2480 ms，就说明 Tomcat 成功运行了，当然 ms 数针对每台计算机是不同的，千万不要关闭这个黑色窗体，关闭了 Tomcat 就不再运行了。为什么？还记得我们所编写的服务器端 Socket 程序吗？这个 Tomcat 就是那个程序，现在程序在 8080 端口监听着，为什么不是 80 端口呢？咱们这不是在做实验吗？默认的 Tomcat 实验端口是 8080。

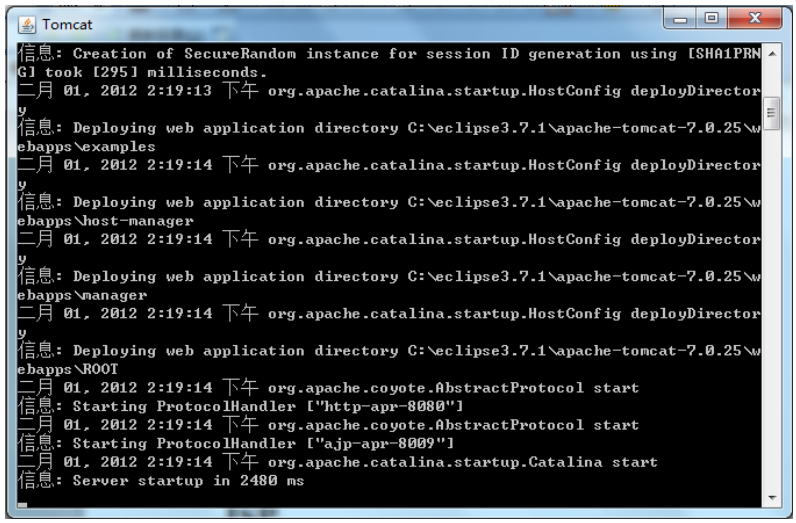


图 1-5

我们要验证一下 Tomcat 运行了，打开 IE 浏览器，输入 URL：http://127.0.0.1:8080，这个 127.0.0.1 是本机的回传地址，通过:8080 来指定 80 以外的端口号，至于“http://”是什么意思回头再解释。

如果你能够在浏览器中看到如图 1-6 所示的画面，说明 Tomcat 是工作的，当然不同版本的 Tomcat 提供的页面是不同的，不过我想这只小猫将一直存在。

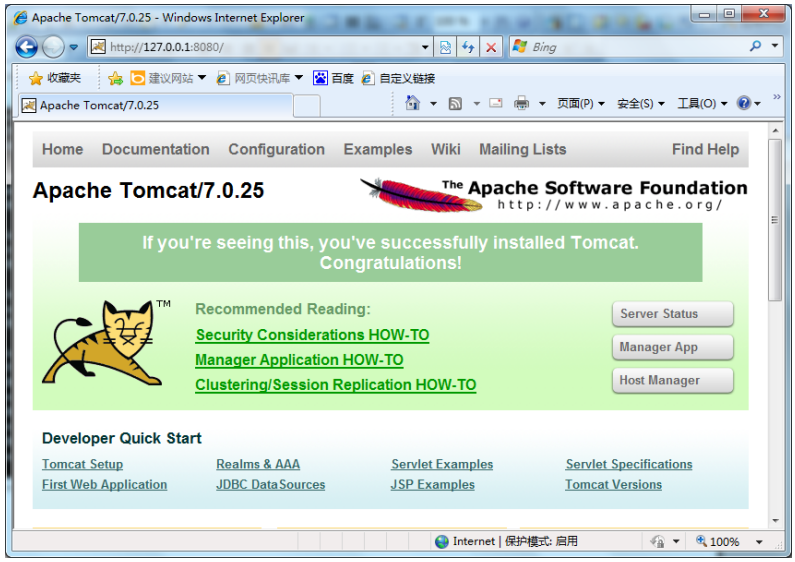


图 1-6

现在访问的自然是 Tomcat 所提供的默认网页，我们是希望能够提供我们所编写的网页，还记得前面我描述的 Web 访问的工作过程吗，如果我们请求的网页是 aaa.html，那么 Tomcat 得到

这个请求后，会到自己所管理的目录中寻找这个文件，注意我所使用的语言，是到自己所管理的目录中，而不是在计算机的硬盘上，当然 Tomcat 所管理的目录也在硬盘上，但是它不能肆无忌惮地访问整个硬盘，Tomcat 所管理的目录就是我们运行的 Tomcat 目录中的 webapps 目录。

我们能够看到在这个 webapps 中有一个子目录的名字是大写的 ROOT 目录，这是默认的根目录，我们能够看到这个目录中已经有了一些文件，这些文件就是上面那个有小猫的网页文件。我们还看到其中有一个目录叫做 WEB-INF，这是作为 Tomcat 要求的，后面我们会频繁地接触到这个目录，现在还不需要做什么。如图 1-7 所示。

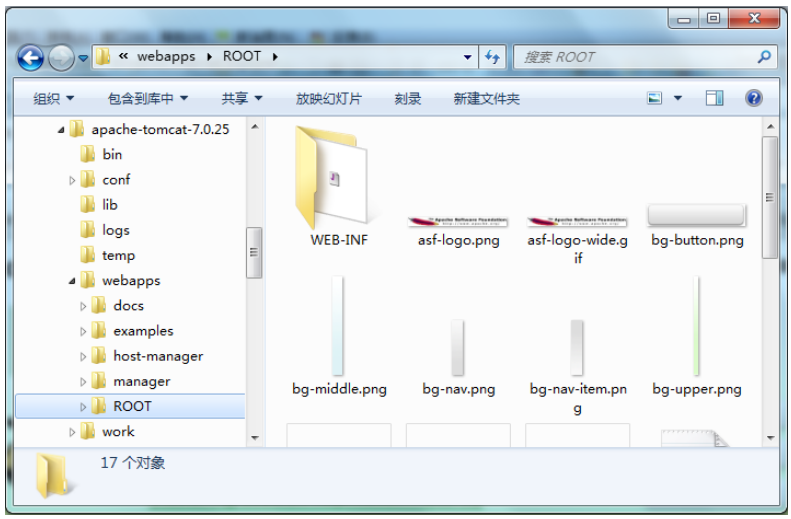


图 1-7

为了验证这些说法，我在 ROOT 目录中新建一个文本文件叫做 aaa.html，注意要避免计算机自动给你提供.txt 这个扩展名。然后我们在这个文件中输入一句话：“这是我编写的第一个网页”，输入后保存。

我们再到浏览器那里输入 URL：http://127.0.0.1:8080/aaa.html，然后回车，你的浏览器上有这句话了吗？如果没有就在重复我提供的这些过程，将前面提供的那个网页访问的示意图套到这个应用的例子中，如图 1-8 所示。

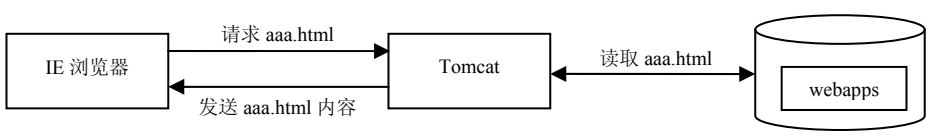


图 1-8

1.4 MyTomcat

有人不相信我所描述的这个过程，如果你有这个念头，我觉得你会是一个有前途的程序员，

你不应该相信任何人或文档上面对原理的描述，除非你自己亲身验证过，这么说不仅仅是因为在计算机行业里没有任何权威，有的时候同样看到听到的话，两个人的理解会有很大的不同。

你们现在完全有能力用程序来验证我所描述的 Web 访问流程，你可以关闭掉 Tomcat，自己做一个假的 Tomcat，然后让真的 IE 浏览器进行同样的访问，看看会不会有 aaa.html 这样的字符串发送给 Tomcat。

整理一下思路，真正的 Tomcat 会做什么事情呢？Tomcat 会监听 8080 端口，当用户启动浏览器，输入网址访问到你的计算机和定义好的 8080 端口时，我们的程序会获得一个服务器端的 Socket，然后获取这个 Socket 的输入流，这样我们就可以通过这个输入流接收用户浏览器发送过来的信息，目前我们好奇的是浏览器会发什么？下面的工作我们先不考虑，现在来看看是否能够接收到请求。

我们来编写一个叫做 MyTomcat 的程序。

```
/*
 *
 * 伪装的 Tomcat，来验证 Web 访问流程
 *
 */
import java.net.*;
import java.io.*;

public class MyTomcat {
    public static void main(String args[]){
        try{
            ServerSocket ss = new ServerSocket(8080);

            System.out.println("Server startup in 1428ms");
            Socket s = ss.accept();

            InputStream is = s.getInputStream();
            InputStreamReader isr = new InputStreamReader(is);
            BufferedReader br = new BufferedReader(isr);

            System.out.println(br.readLine());
        }catch(Exception e){}
    }
}
```

上述程序对于经过 Java 学习的学生应该相当简单，运行的时候千万不要忘记，要关闭真的 Tomcat，否则将发生 8080 端口被占用的异常。运行这个程序后，控制台上将显示 Server startup in 1428ms 这句话，这纯粹是为了让假的 Tomcat 尽可能的像真的。然后到浏览器那里单击“刷新”按钮，回到控制台，你将看见这样一句话：GET /aaa.html HTTP/1.1。

里面有一个部分是/aaa.html，这就是我所说的请求，字符串，还有别的信息，前面有一个单词 GET，你要记住这个单词，后面的学习会用到。最后是 HTTP/1.1，我在测试 Tomcat 有没有工

作的时候用到过这个 HTTP，当时我将这个单词的解释留了下来，别急，我现在还不解释。

多出来的这些信息让我有了一个想法，或许会有更多的信息存在，我将输入流读取的语句多复制了一些，我们再来试试。

```
/*
 *
 * 伪装的 Tomcat，来验证 Web 访问流程
 *
 */
import java.net.*;
import java.io.*;

public class MyTomcat {
    public static void main(String args[]){
        try{
            ServerSocket ss = new ServerSocket(8080);

            System.out.println("Server startup in 1428ms");
            Socket s = ss.accept();

            InputStream is = s.getInputStream();
            InputStreamReader isr = new InputStreamReader(is);
            BufferedReader br = new BufferedReader(isr);

            System.out.println(br.readLine());
            System.out.println(br.readLine());
            System.out.println(br.readLine());
            System.out.println(br.readLine());
            System.out.println(br.readLine());
            System.out.println(br.readLine());
            System.out.println(br.readLine());
            System.out.println(br.readLine());
            System.out.println(br.readLine());
            System.out.println(br.readLine());
            System.out.println(br.readLine());
            System.out.println(br.readLine());
            System.out.println(br.readLine());
        }catch(Exception e){}
    }
}
```

不出所料，刷新浏览器的时候，出现了大量的信息被打印出来，如下所示。

```
Server startup in 1428ms
GET /aaa.html HTTP/1.1
Accept: */*
Accept-Language: zh-CN
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1;
Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR
```

```
3.0.30729; .NET4.0C; Media Center PC 6.0; MALC)
Accept-Encoding: gzip, deflate
If-Modified-Since: Wed, 01 Feb 2012 06:41:26 GMT
If-None-Match: W/"22-1328078486338"
Host: 127.0.0.1:8080
Connection: Keep-Alive
```

这说明访问的过程比我们想象的要复杂多了，我来解释一下这些信息，GET /aaa.html HTTP/1.1 这句话已经解释过了。

Accept: */*，是指这个浏览器能够接收的信息类型。现在是浏览器将一些信息发送到 Tomcat，那么我们的浏览器能够接收*/*，*在计算机里通常是通配符，这样写意味着可以代替任意字符，如果不是用通配符，可能的写法是 text/html，意思是能够接收基于文本的 html 内容，我们有时访问一个网站时，会遇到内容以接近于 Word 的形式显示，这是因为这个网站提供的文件就是 Word 格式的文件，浏览器本身是不能解释 Word 格式的，在你的浏览器中能够显示这样的文件，是因为你的电脑中安装了 Word，这个 Word 程序扩展了浏览器的功能，如果这个浏览器不使用通配符，就会在这个地方告诉 Tomcat，它有看 Word 文件的能力。

Accept-Language: zh-CN，这句话告诉 Tomcat，客户计算机的操作系统是中文的，我们有这样的体验，在浏览器中输入 www.google.com，你看到的是中文的网页，想象一下，如果是美国人输入完全相同的 URL，应该看到英文网页，同样的访问，不同国家的人能够得到不同语言的网页，靠的就是浏览器提供过服务器的这个信息。

User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; .NET4.0C; Media Center PC 6.0; MALC)，这是一句话，因为没法在控制台的一行显示，所以被自动换行了，这句话告诉 Tomcat 浏览器是什么，操作系统是什么，支持的其他技术框架是什么，当然每个人由于计算机安装的软件不同，这行信息是不同的，其中有些信息显示的不是在公开市场上的名字。有时能够在一些网站上看到显示着你的计算机的信息，这会让人吓一跳，其实是你的浏览器告诉服务器的。

Accept-Encoding: gzip, deflate，我们知道在网络上传送压缩文件是合理的，对于计算机来说网络是一个很慢的通路，但是网页文件不是压缩的，我们当然可以在服务器上将网页文件压缩了再传送，这样网页打开的效率就会高很多，问题是我们并不知道用户的计算机有没有解压缩的能力，我的浏览器告诉 Tomcat 它能够打开 gzip 格式的压缩文件。

If-Modified-Since: Wed, 01 Feb 2012 06:41:26 GMT ; If-None-Match: W/"22-1328078486338"，这两句话要和在一起看，你能理解第一句话主要包含了时间的信息，而第二句话好像是一个随机生成的编码吗？在很多场合下都有这样的应用，我们管这个应用叫做“时间戳”，用处是当你访问了一次服务器后，再次访问时服务器就知道你是谁，你来过，服务器会保留你的一些信息，并且和其他人区分开。过去我们没有遇到这样的问题，原因是过去我们的 Socket 应用是面向稳定连接的，就是说两台计算机通信的过程中，网络是不能断掉的，这样服务器自然有办法区分不同的客户端了，但是浏览器的访问方式是面向不稳定连接的，也就是说在浏

览器不需要的时候，网络是可以断开的，再次访问意味着重新建立连接，这样有些时候我们就要利用这个“时间戳”来与其他访问区分开。你能够理解在浏览器第一次访问服务器的时候，它是没有“时间戳”信息的吗？这个信息应该是 Tomcat 生成并发送给浏览器的，以后每次访问，浏览器都会将这个“时间戳”传给服务器，我们清除一下浏览器，再试试看。

Host: 127.0.0.1:8080，这个不需要太多解释，就是告诉 Tomcat，我当时的访问的主机地址和端口号。

Connection: Keep-Alive，这里说明了浏览器和 Tomcat 之间的连接模式，虽然之前提到过，这是面向不稳定连接的应用，需要时就连接，用完了就断开，但是很明显在大多数情况下不需要这样做，这样做的效率很低，所以在条件允许的情况下，浏览器和 Tomcat 约定了 Keep-Alive，这样就不用频繁地建立连接，再断开连接了，如果对这个具体的连接属性有兴趣，网络上到处都是详细的解释。

我们简单地了解到浏览器在请求网页的时候，都发送了什么到 Tomcat，你有没有意识到，我们用这种方式将浏览器发送的信息“骗”了出来，如果我们写出来一个假的 IE，也将这些信息发送出去，那么真的 Tomcat 岂不是会认为我就是一个 IE 浏览器，岂不是也能将网页发送给我们。

这么做或许会被别有用心的人所利用，比如现在有一个投票的网站，通常的作弊方法是不断地点击给某个人投票，现在我们可以写个程序，伪装成浏览器，用循环来发送投票的字符串，我就是随便说说，你做什么不是我能决定的，至少抢火车票的程序没有过错。

当然现在的网站在这种地方使用验证码，验证码是个图片，每次投票都需要附加这个验证码信息给服务器比对，识别图片上信息的过程需要人来参与，在很大程度上阻止了这个作弊程序。

你现在能够自己尝试着编写出来 MyIE 了吧！如果没有这个信心，还是建议你熟练掌握我的上一本关于 Java 编程的书。在尝试前，记得到浏览器中找工具菜单里面的 Internet 选项，将所有的历史记录都删除掉，然后运行假的 Tomcat，以及真的 IE 再“骗”回来没有“时间戳”的内容，这样 MyIE 更容易成功。

1.5 MyIE

```
/*
 *
 * 伪装的 IE，来验证 Web 访问流程
 *
 */
import java.net.*;
import java.io.*;

public class MyIE {
    public static void main(String args[]) {
        try {
```

```
            // 真正的 IE 连接信息是通过对 URL 的字符串拆分得到的
            Socket s = new Socket("127.0.0.1", 8080);

            OutputStream os = s.getOutputStream();
            OutputStreamWriter osw = new OutputStreamWriter(os);
            PrintWriter pw = new PrintWriter(osw, true);

            pw.println("GET /aaa.html HTTP/1.1");
            pw.println("Accept: */*");
            pw.println("Accept-Language: zh-CN");
            pw.println("User-Agent: Mozilla/4.0 (compatible; MSIE 8.0;
Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR
3.5.30729; .NET CLR 3.0.30729; .NET4.0C; Media Center PC 6.0; MALC)");
            pw.println("Accept-Encoding: gzip, deflate");
            pw.println("Host: 127.0.0.1:8080");
            pw.println("Connection: Keep-Alive");
            pw.println("");// 需要一个空行结束，才能得到 Tomcat 回传的信息
            //接收 Tomcat 传送回来的信息
            InputStream is = s.getInputStream();
            InputStreamReader isr = new InputStreamReader(is);
            BufferedReader br = new BufferedReader(isr);

            System.out.println(br.readLine());
            System.out.println(br.readLine());
            System.out.println(br.readLine());
            System.out.println(br.readLine());
            System.out.println(br.readLine());
            System.out.println(br.readLine());
            System.out.println(br.readLine());
            System.out.println(br.readLine());
            System.out.println(br.readLine());
            System.out.println(br.readLine());
            System.out.println(br.readLine());
        } catch (Exception e) {}
    }
}
```

代码并不困难，传出的信息记得要加上一个空行 Tomcat 才会认，现在要运行真的 Tomcat，然后再运行程序，看看得到了什么？依然是一大堆信息，要等很长时间才能看见网页中的那句话。

下面就是我得到的信息，你的应该是大同小异的。

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Accept-Ranges: bytes
ETag: W/"22-1328078486338"
Last-Modified: Wed, 01 Feb 2012 06:41:26 GMT
Content-Type: text/html
Content-Length: 22
```

```
Date: Thu, 02 Feb 2012 02:56:12 GMT
```

这是我编写的第一个网页。

我们还是一行行地来分析。

HTTP/1.1 200 OK，我已经忍不住要告诉你 HTTP 是什么了，我一直在说浏览器将什么信息发送给了 Tomcat，你想想你的计算机上的浏览器会知道网站服务器用的是什么软件吗？其实浏览器才不管服务器是 Tomcat 还是 IIS，它只要将自己该发送的信息发送出去就好了，Web Server 也要遵守彼此的约定，这样的约定并不是 IE 和 Tomcat 协商的结果，应该是天下的浏览器和天下的 Web Server 都认同的约定，这样互联网才能有效地运作。

这样的约定被称为协议，我们现在看到的协议叫做 HTTP 协议，Tomcat 告诉 IE 现在它所使用的版本是 1.1。

200 OK，OK 不用解释，就是没问题的意思，200 也没有问题，说明这个网页正常的传送到了浏览器端，如果不是 200，你上网的时候有可能会遇到 404 或 500，这两个数意味着出问题了，当然还有其他数字的含义，我不做过多的介绍，有兴趣的话，可以上网去查。

Server: Apache-Coyote/1.1，这句话的意思是服务器软件是什么，我们并没有得到期待中的 Tomcat，但是至少知道这是 Apache 的 Web Server。

Accept-Ranges: bytes，这句话告诉客户端该服务器是否支持断点续传，bytes 意味着支持。

ETag: W/"22-1328078486338"；Last-Modified: Wed, 01 Feb 2012 06:41:26 GMT 这两句话前面有所涉及，这是 Tomcat 为这个浏览器生成的“时间戳”。

Content-Type: text/html，还记得浏览器告诉服务器能够接受*/*，服务器现在告诉浏览器，我这次传送的内容是基于文本文件的 html 格式。

Content-Length: 22，内容的长度是 22 个字节。

Date: Thu, 02 Feb 2012 02:56:12 GMT，时间信息。

然后就是网页的具体内容。

1.6 再谈 Tomcat

当然完全编写 Tomcat 要比我的这个实验复杂得多，能够想象一个 Tomcat 将会应对很多浏览器的访问，如果你能够熟练掌握即时聊天程序，其实那个服务器代码和 Tomcat 很相似，不过可能要多出来对 HTTP 协议的支持，支持多个用户访问，即多线程操作。

学习 Java Web 和学习 Java 的难点完全不同，Java 再难，所有的代码都是你一点点的编写出了的，对于自己编写的代码，我们会心里感到踏实，而 Java Web 的代码要和这个 Tomcat 打交道，这意味着有些代码是你编写的，运行起来就跑到 Tomcat 里面去了，很多人面对这样的情况，适应不了，心里不踏实，这要求你能够猜出这个时候 Tomcat 做了什么，如果你能做到这

点，Java Web 的学习就变得相当简单了。

最完美的做到这点的方式是，干脆我们真正写出来一个 Tomcat 好了，当然这是个艰巨的挑战，但是如果你这样做了，即便最后失败了，你对 Java Web 的理解也会远远超过照葫芦画瓢的结果。再说理论上写出 Tomcat 的基本功能并不是不可能，建议你随着学习的深入，每当我讨论到 Tomcat 所做的事情时，至少写代码来模拟这些功能，这样我们就站在 Tomcat 的高度来学习 Java Web，而不仅仅是讨论 Servlet 或是 JSP 怎么写。当然因为 Tomcat 是源代码公开的软件，你可以到网站上下载它的源代码，读一读非常有好处。

第 2 章

学习 HTML

我们现在先来解决一个相对容易的部分，学习 HTML，很多人对于这个部分嗤之以鼻，因为 HTML 太普遍了，加上 HTML 在设计之初就是为了容易编写，所以很多人都会 HTML，不就是有些规定的文本文件吗？问题是会 HTML 的人中，有很多一直在使用 HTML 编辑工具，比如著名的 Dreamweaver，那么我认为你有一定的基础，但是对于这本书的要求是不够的，我们学 习 HTML 的目的是为了将 HTML 的代码嵌入到 Java 语言中，所以能够纯粹的手写 HTML 代码是基本的要求。也因为这个原因，在本书中，我不会太纠缠于 HTML 的细节，特别是关于美化方面的内容。

另外不知道我所理解的 HTML 和你已经掌握的 HTML 是否一样，我所认为的 HTML 知识严格说应该叫做前段技术，包含 HTML、CSS、JavaScript，这些技术并不像想象的那么容易，到目前为止，我教的最好的学生毕业后一直从事着前段技术的工作，目前带领着 100 多人的开发团队，都是玩 HTML 的。

2.1 认识 HTML

现在我们先来认识一下什么是 HTML，先来看 HT，意思是超文本，我个人觉得就这本身就是一场革命，我们知道过去看书的方式，你只能一行行一页页的读下去，如果遇到有兴趣的内容，要自己想办法在书上标记好，暂停这次阅读，回头再继续，内容的组织形式是线性的，阅读也只能是线性的。有了超文本后，有兴趣的地方可能会有超链接，你可以先看感兴趣的，看完后也可以很方便的回到原来跳过的地方，继续阅读。对于很多人来说，我所谓的革命很矫情，因为很多年轻人没有经历这个变化过程，所以干脆不用。超文本就是有超链接的导航能力的内容。

ML 是标记语言的意思，什么是标记语言呢？我们通常在电脑里会用到 Word，在 Word 中如果选中了一段话，然后单击加粗按钮，这段话就会变成粗体的，这个所见即所得的能力让微软在一段时间里都引以为豪，我们深入去思考，在计算机里存储这个文件，是否会存储加粗的文字，答案是不可能，因为文件不仅仅有加粗，文字有各种各样的样式，除非用图像来存储，可是图像又太占地方了，其实我们存储的往往是文字的编码，事实上加粗是 Word 在这段话的开头和结尾

加上了特定的标记，Word 这个软件打开一篇文档时，会扫描这些特定的标记，一旦遇到便会显示成粗体。

问题是 Word 定义的标记你看不到，它根本就不是文本字符，这些标记的存在意味着用 Word 编辑的文件也不是文本文件，其他程序如果要访问 Word 文件会很困难，微软当然喜欢这样，只有微软知道标记这阻止了别人的程序进入这个领域，但是这种隐含的标记也阻碍了 Word 文件的通用性。

能不能在对文件进行标记的同时，将标记定义成任何程序都能理解的文本，HTML 就定义了这样的标记，这些标记形成了标记语言。

使用文本有个问题，就是标记有可能与真正网页的内容冲突，HTML 用<>来区分什么是标记、什么是内容，在 HTML 中最重要的标记是<html></html>，你注意到我提供了一对 html 标记，都放在尖括号中，后面的多出来一个/，这是为了说明开始和结束，每个网页文件都应该以<html>为开头，</html>为结尾。

事实上我们前面已经创建了一个 HTML 文件，还记得 aaa.html 文件吗，HTML 文件以.html 为扩展名，也可以使用.htm，这是因为在 HTML 刚被发明的时期，还有很多电脑上的操作系统最多只支持三个字母的扩展名。

可是在之前的那个 aaa.html 文件里并没有<html></html>标记呀，这是因为浏览器太宽松了，虽然没有提供 html 标记，浏览器也能识别代码，但是这里强烈建议写全这样的标记，因为有很多浏览器存在，每一种浏览器的宽松程度是不同的，尽可能的遵守 HTML 的规则能够最大程度的确保不同浏览器的显示效果一致。

还要说明一点，在 HTML 的规范中，标记是不分大小写的，但是同样强烈建议写成小写，因为下一代的 HTML 都是要求小写的，无论是 W3C 组织定义的 HTML4，还是 XHTML。

我们创建好了 aaa.html 文件后，双击文件，计算机里默认的浏览器就会启动，并且显示这个网页文件，我们把这个过程称之为浏览器读取了硬盘上的一个 html 文件。可是之前不是这样做的，我们是启动了 Tomcat，然后用浏览器访问 Tomcat 的 IP 地址和端口号，最终得到这个 html 文件，这两种形式的访问是不同的，在现在这个阶段没什么差别，但是很快你就会发现，我们需要通过 Tomcat 来得到网页内容。

2.2　做百度的首页

百度首页相比大家都比较熟悉了，这是我能够找到的最简单的网页，其实这个网页并不像看起来那么简单，我们不管网页背后的功能，先照葫芦画瓢来做出这个网页的样子。如图 2-1 所示为百度首页。

图 2-1

从上向下看，你发现浏览器的标题变成了“百度一下，你就知道”，我们来实现这个功能，需要一个完整的 HTML 文件结构了。

```
<html>
    <head>
        <title>百度一下，你就知道</title>
    </head>

    <body>
    </body>
</html>
```

整个 HTML 文件的内容都被放到 html 标记中，在 html 标记里面有两个标记，一个是 head，一个是 body，body 中的内容将被显示在浏览器中间显示内容的区域里，而 head 通常放显示内容之外的代码。

在 head 中有一个标记 title，用来定义显示这个网页的浏览器标题，就是浏览器窗体最上边蓝色区域的文字。

上面的代码的显示如图 2-2 所示，请注意标题的变化。

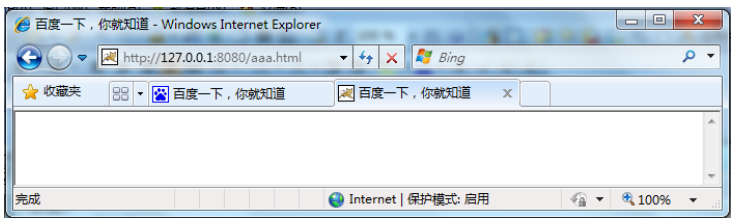


图 2-2

虽然代码简单，但是希望你不要掉以轻心，所有的代码都要亲自敲出来，不能觉得看得懂就完了，最好是多敲几遍，心里有什么想法就敲出代码试一下。

在浏览器的内容显示区域，第一行是三个超链接搜索设置、登录和注册，超链接是 HTML 中最重要的能力，正是因为有了超链接，我们才称它是超文本的技术，超链接的标记是 a，但是我们知道当网页上有超链接的话，单击这个超链接，网页就会跳转到相应的位置，跳转到什么地方，我们是通过 a 这个标记的属性来提供的。

因为在这个阶段我只想实现百度的首页，不打算完全做出来一个百度那样的网站，所以其中的超链接都借助于百度公司的其他网页。看下面的代码。

```
<html>
    <head>
        <title>百度一下，你就知道</title>
    </head>

    <body>
        <a href="http://www.baidu.com/gaoji/preferences.html">搜索设置</a>
        <a href="http://passport.baidu.com">登录</a>
        <a href="https://passport.baidu.com">注册</a>
    </body>
</html>
```

超链接的属性是 href，你会发现我将单击后需要链接到的 URL 放到了这个属性中，这个代码的显示效果如图 2-3 所示。

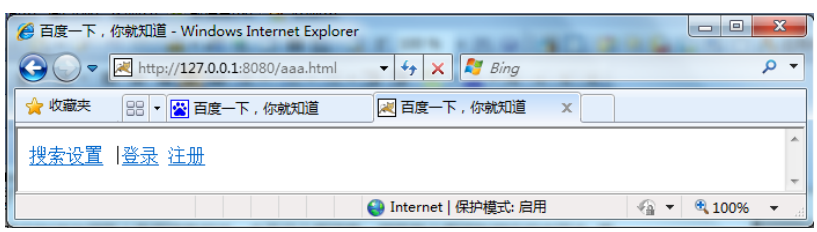


图 2-3

和百度的页面相比有几个问题，原网页的位置是靠右的，字体比这个小，颜色比这个要蓝。

传统的靠右的写法是 ，字体大小和颜色是 font 标记中的两个属性。改造后的代码如下。

```
<html>
    <head>
        <title>百度一下，你就知道</title>
    </head>

    <body>
        <p align="right">
            <font color="blue" size="2">
                <a href="http://www.baidu.com/gaoji/preferences.html">搜
```

```
索设置</a>
                <a href="http://passport.baidu.com">登录</a>
                <a href="https://passport.baidu.com">注册</a>
            </font>
        </p>
    </body>
</html>
```

这里面有几点要注意，标记 本身是有功能的，标记 会产生分段的效果，如果是紧密排列的内容，使用了标记 ，这段文字的前后会有很大的间隔。

标记和标记之间是不能嵌套的，不能写成这个样子 ，颜色上如果你觉得 blue 还不是的话，可以使用 color="0000ff"这样的形式设置颜色，这是 RGB 模式的颜色编号顺序，每两位是一个颜色，最小 00，最大 ff，这样你就能调配出所有的颜色了。

上述代码的效果如图 2-4 所示。

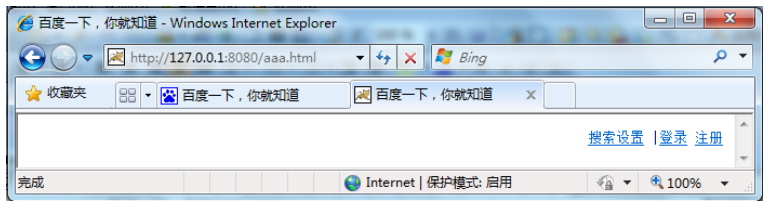


图 2-4

我们再往下看，下面有一个居中的百度的商标。如图 2-5 所示。

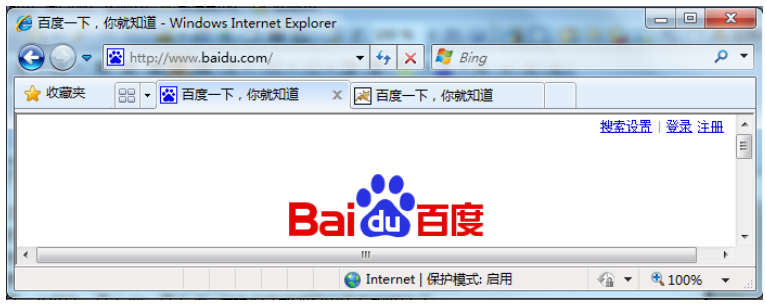


图 2-5

这是一个图片，按说我们要画一个这样的图片，但是这不是我擅长的事情，我先借用一下这个图片，用你的浏览器访问百度的首页，然后用鼠标右键单击这个图片，你将看到在右键菜单有一个“图片另存为...”选项，选择它，会弹出一个对话框，问你要将这个图片存在什么地方，我们将它存到我们所编写的网页的同一个目录中，命名为 baidu.gif。

将图片放到网页中使用以下标记，这个标记有点特别的地方，它不需要什么内容，对于不包含内容的标记，我们在结尾加上/，相当于<img></img>。

标记能够让内容居中，当然你也可以用 ，因为上一个部分的内容

被标记在<p>中，所以上面的内容和这个图片会自然而然进行换行。

```
<html>
    <head>
        <title>百度一下，你就知道</title>
    </head>

    <body>
        <p align="right">
            <font color="blue" size="2">
                <a  href="http://www.baidu.com/gaoji/preferences.html">搜
索设置</a>
                <a href="http://passport.baidu.com">登录</a>
                <a href="https://passport.baidu.com">注册</a>
            </font>
        </p>
        <center>
            <img src="baidu.gif"/>
        <center>
    </body>
</html>
```

上述代码的效果如图 2-6 所示。

图 2-6

继续向下能够看见一排超链接，我们照着完成就好了。

```
<html>
    <head>
        <title>百度一下，你就知道</title>
    </head>

    <body>
        <p align="right">
            <font color="blue" size="2">
                <a  href="http://www.baidu.com/gaoji/preferences.html">搜
索设置</a>
                <a href="http://passport.baidu.com">登录</a>
```

```
            <a href="https://passport.baidu.com">注册</a>
        </font>
    </p>
    <center>
        <img src="baidu.gif"/>
    <center>

    <font color="blue" size="3">
        <a href="http://news.baidu.com">新 闻</a>
        <font color="black"><b> 网 页</b></font>
        <a href="http://tieba.baidu.com"> 贴 吧</a>
        <a href="http://zhidao.baidu.com">知 道</a>
        <a href="http://mp3.baidu.com">MP3</a>
        <a href="http://image.baidu.com">图 片</a>
        <a href="http://video.baidu.com">视 频</a>
        <a href="http://map.baidu.com">地 图</a>
    </font>
</body>
</html>
```

因为其中网页的那个不是超链接，所以它是黑色加粗的，黑色的设置应该没问题，加粗的标记是<b>，我们通常将加粗和斜体<i>、下划线<u>一起记，不过你发现超链接是自动就有下划线的，如图 2-7 所示。

图 2-7

再向下就是大家熟知的输入框和按钮了，输入框和按钮在网页设计里面是一个特别的部分，对于我们来说尤为重要，我们知道学习 HTML 的主要目的是为了将浏览器作为程序的界面，我们不是为了成为一个网页设计师，网页的漂亮固然重要，但是制作程序界面，我们是需要能够得到用户的输入，所以将来我们将不断的和输入框打交道。

输入的标记是<input>，其实输入有很多种类型，输入框是输入，按钮也是输入，还有单选、多选之类的，这些都是使用<input>标记，但是仅仅知道如何产生输入框还不行，用户在输入框中输入了内容，然后单击按钮，是不是应该有一个程序来得到并处理用户所输入的内容呢，

就目前来看，根本不敢想象 HTML 能够处理信息，因为这个语言没有变量、没有循环、没有判断，要处理信息还是要找 Java 程序，而 Java 程序并不是和网页绑定在一起的，通常的结构是用户的浏览器一端是 HTML，Java 程序在服务器的 Tomcat 里，它的形式变成了 JSP。

那么我们在定义输入框的时候，就要指定由哪个 JSP 程序来处理输入框中的内容，像百度首页就一个输入框这么简单的网页并不多，通常会是一组输入框，所以我们将一组输入框放到一个叫做<form>的标记中，并且在这个标记的 action 属性中指定处理程序。

这个部分的内容被称为表单，由于<form>标记非常重要，所以人们通常将这个标记称为 form 表单。

下述代码实现了这一步。

```
<html>
    <head>
        <title>百度一下，你就知道</title>
    </head>

    <body>
        <p align="right">
            <font color="blue" size="2">
                <a  href="http://www.baidu.com/gaoji/preferences.html">搜
索设置</a>
                <a href="http://passport.baidu.com">登录</a>
                <a href="https://passport.baidu.com">注册</a>
            </font>
        </p>
        <center>
            <img src="baidu.gif"/>
        <center>

        <font color="blue" size="3">
            <a href="http://news.baidu.com">新 闻</a>
            <font color="black"><b> 网 页</b></font>
            <a href="http://tieba.baidu.com"> 贴 吧</a>
            <a href="http://zhidao.baidu.com">知 道</a>
            <a href="http://mp3.baidu.com">MP3</a>
            <a href="http://image.baidu.com">图 片</a>
            <a href="http://video.baidu.com">视 频</a>
            <a href="http://map.baidu.com">地 图</a>
        </font>
        <form action="bbb.jsp" >
            <input type="text" name="wd" size="50">
            <input type="submit" value="百度一下">
        </form>

    </body>
</html>
```

在这个代码中，我假设输入框的处理程序是 bbb.jsp，输入框的类型是 text，这个输入框必须有个名字，否则后台的处理程序就没办法找到这个输入框中的值了，按钮的类型是 submit，事实上有几个类型的样子都是按钮，submit 的按钮专门负责将用户的输入送到 bbb.jsp 去，value 属性是按钮上面显示的文字。效果如图 2-8 所示。

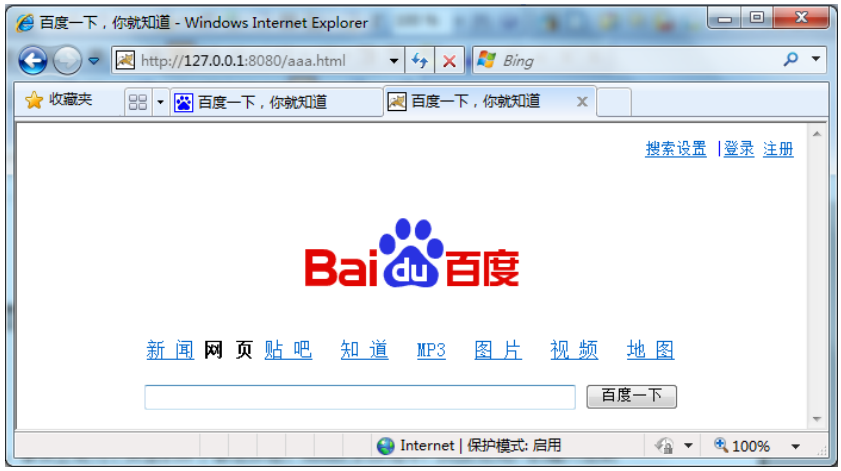


图 2-8

你发现这个效果并不完美，输入框和按钮比百度的要小很多，现在这个阶段，我们先忽略这个问题，以后再想办法做得更像。

百度首页下面的内容，我不再讲解了，你应该完全有能力自己完成，我将全部的代码提供给你，务必自己尝试后，再看我的代码。

```
<html>
    <head>
        <title>百度一下，你就知道</title>
    </head>

    <body>
        <p align="right">
            <font color="blue" size="2">
                <a href="http://www.baidu.com/gaoji/preferences.html">搜
索设置</a>
                <a href="http://passport.baidu.com">登录</a>
                <a href="https://passport.baidu.com">注册</a>
            </font>
        </p>
        <center>
            <img src="baidu.gif"/>
        <center>

        <font color="blue" size="2">
```

```
            <a href="http://news.baidu.com">新 闻</a>
            <font color="black"><b> 网 页</b></font>
            <a href="http://tieba.baidu.com"> 贴 吧</a>
            <a href="http://zhidao.baidu.com">知 道</a>
            <a href="http://mp3.baidu.com">MP3</a>
            <a href="http://image.baidu.com">图 片</a>
            <a href="http://video.baidu.com">视 频</a>
            <a href="http://map.baidu.com">地 图</a>
        </font>
        <form action="bbb.jsp" >
            <input type="text" name="wd" size="50">
            <input type="submit" value="百度一下">
        </form>
        <font color="blue" size="2">
            <a href="http://hi.baidu.com">空间</a>
            <a href="http://baike.baidu.com">百科</a>
            <a href="http://www.hao123.com">hao123</a>
            <a href="/more/">更多&gt;&gt;</a>
        </font>
        <br/><br/><br/><br/><br/><br/>
        <font color="blue" size="2">
            <p><a href="http://utility.baidu.com">把百度设为主页</a>
            <a href="add.jsp">把百度加入收藏夹</a></p>
            <p><a href="http://e.baidu.com/?refer=888">加入百度推广</a>
            <a href="http://top.baidu.com">搜索风云榜</a>
            <a href="http://home.baidu.com">关于百度</a>
            <a href="http://ir.baidu.com">About Baidu</a></p>
            <p>&copy;2012 Baidu <a href="/duty/">使用百度前必读</a>
            <a href="http://www.miibeian.gov.cn">京 ICP 证 030173 号</a>
            <img src="http://gimg.baidu.com/img/gs.gif"></p>
        </font>
    </body>
</html>
```

上述代码大部分都应该能够看懂，有几个地方要解释一下，在更多的后面有两个大于号，要知道定义标记所使用的尖括号，事实上是由大于号和小于号构成的，内容里面真正要显示大于号或小于号时，是有可能和标记定义发生冲突的，为此 HTML 用一个叫做实体的东西来代替，实体的定义是&开头，;结尾，这里面用的>就是大于号的实体，小于号是<，另外一个常用的实体是 ，这个是空格，因为在 HTML 文件中再多的空格和换行显示的时候都只是一个空格。如果要显示很多空格，就要用很多 才有效，©也是一个实体，这本书不是词典，并不列举所有的实体，如果你真的有兴趣或是需要的话，可以自己到网上查。

换行也是个问题，要想在页面上有一个换行，就要写
标记，其实写
也行，只是这样并不规范。

最终的效果和百度的页面虽然已经很像了，但是还是有所差别的，看来我们要学习更多的 HTML 技术，才能真正随心所欲。

2.3 搜狐邮箱的用户登录

用户登录非常重要，作为 Java Web 的程序员，几乎任何一个项目都难以逃脱这个页面。为了便于学习，我寻找了很多网站的用户登录界面，尽力剔除过于复杂和远超过目前技术的页面，最终京东商城和搜狐邮箱的用户登录符合我的要求，比较这两个页面，我个人认为搜狐邮箱的用户登录页面相对漂亮一些，当然如果你喜欢京东商城的用户登录，完全可以顺着我的描述自行完成，我十分赞同在我的任务外，你自己寻找相似的任务去进行练习，所有遇到的困难都将帮助你学到更多。

在你的浏览器中输入 mail.sohu.com，你将看到这个用户登录页面。如图 2-9 所示。

图 2-9

浏览器的标题，上下的图片，以及左边的文字我想你都不会有太大问题，我们暂时将其放到一边，先专心完成主要的表单，表单界面如图 2-10 所示。

这是页面中的一个部分，我们先来实现这个部分。

在网页上用鼠标右键单击这个区域，右键菜单中会出现“背景另存为...”选项，选择后在保存的对话框中将这个图片存入 HTML 文件的目录中，命名为 pic_login.gif，得到这个图片后，我们将其设置成网页的背景图片。

图 2-10

```
<html>
    <head>
        <title>搜狐闪电邮箱——十年专业品质，国内一流的免费邮箱</title>
    </head>
    <body background="pic_login.gif">
    </body>
</html>
```

现在用浏览器显示这个网页，会发现出现了好几个这样的背景图片，没办法这个图片太小了，将它作为背景，浏览器只好多显示几个才能铺满，我们将浏览器缩小到恰好这个图片这么大，如图 2-11 所示。

图 2-11

你发现其中的有些内容已经作为图像存在了，我们要做的事情并不太多，下面在这个背景上

填入表单。

```
<html>
    <head>
        <title>搜狐闪电邮箱——十年专业品质，国内一流的免费邮箱</title>
    </head>
    <body background="pic_login.gif">
        <form action="login.jsp">
            邮 箱：<input type="text" name="mail"><br/>
            密 码：<input type="password" name="pass">
            <a herf="repass.html">忘记密码</a><br/>
            <input type="checkbox" name="save">记住登录状态
            <a href="loginhttps.html">https 安全访问</a><br/>
            <input type="submit" value="登录">
        </form>
    </body>
</html>
```

密码的输入框类型是 password，和 text 唯一的区别是在 password 中，用户的输入会被显示成点，这样能够防止用户身边的人偷窥到密码。checkbox 是多选框，如果在一组中有多个多选框，需要将 name 设定成一样的，对应的 radio 是单选。上面代码的效果如图 2-12 所示。

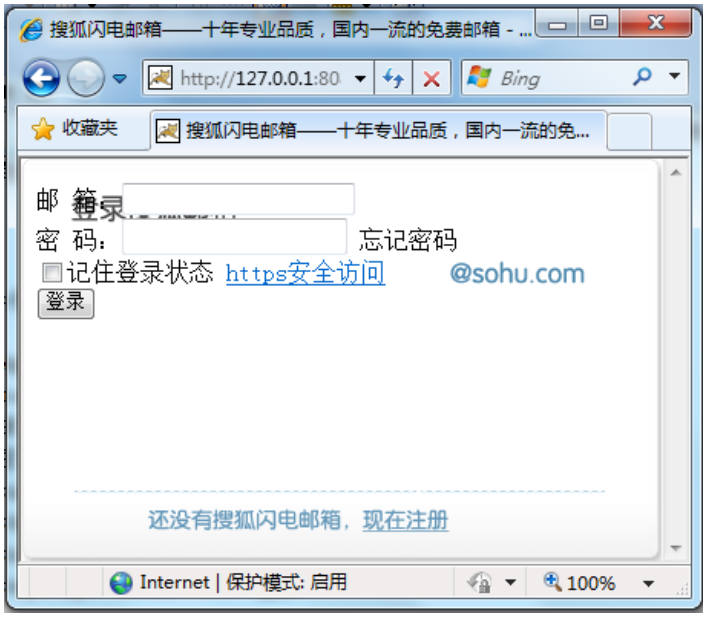


图 2-12

这真是一片混乱呀，看来要做的工作还很多，根本的任务是将上面的这些内容放到合适的位置，这就是网页设计的定位问题。我先通过
的换行，以及 的空格，将这些内容安排到合适的位置，这是一个不断尝试的过程。

```
<html>
    <head>
        <title>搜狐闪电邮箱——十年专业品质，国内一流的免费邮箱</title>
    </head>
    <body background="pic_login.gif">
        <form action="login.jsp">
            <br/><br/><br/>
            <font face="Arial">
            <font size="3">   
            邮 箱：<input type="text" name="mail" size="25"><br/><br/>
               
            密 码：<input type="password" name="pass" size="27">
            </font>
            <font size="2"><a href="repass.html">忘记密码</a><br/>
                      
                  

            <input type="checkbox" name="save">记住登录状态
            <a href="loginhttps.html">https 安全访问</a></font><br/><br/>

            <input type="image" src="button_login.gif"/>

            </font>
        </form>
    </body>
</html>
```

上面代码效果如图 2-13 所示。

图 2-13

在上面的代码中，加粗的部分原本应该是 submit，提交按钮，但为了美观，搜狐用一个图片来代替，图片按钮的类型是 image，至于功能，以后再解决。

如果这个过程你尝试了，就会遇到上下两个输入框长度不一致的问题，无论怎样设置 size，都没办法让它们的长度一样，这是因为 text 和 password 的默认字体是不同的，目前还解决不了这个问题，因为 font 影响不到<input>。

为了定位，我使用了大量的 和
，事实上是将这两个东西作为水平和垂直定位的工具，这并不那么舒服。还有一个大问题需要解决，我怎么将这一堆东西像搜狐的页面一样，放到屏幕的右侧中间的位置，总不能继续大量使用 和
吧。

2.4 京东的购物车

你或许会奇怪，我冷不丁地怎么讲起来表格了，这不像是我的风格呀，这是因为完全做好搜狐邮箱的登录界面，就要将我们之前实现的那些表单移到屏幕的右边，如果使用空格和换行将其在水平和垂直方向定位会非常烦琐，甚至不太现实，所以在很长的一段时间里，人们都是使用表格来进行定位的，先不考虑这个登录界面，我们研究一下表格。

在网页中难免会用到表格，如图 2-14 所示。

商品编号	商品名称	京东价	返现	赠送积分	商品数量	删除商品
10808005	 史蒂夫·乔布斯传（Steve Jobs: A Biography乔布斯唯一正式授权传记简体中文版）	¥51.00	￥0.00	0	1	删除
529236	 微星（msi）Z68S-G43 (G3)主板（IntelZ68 (B3)/LGA1155）	¥599.00	￥0.00	0	1	删除
重量总计：1.25kg　原始金额：￥650.00元 - 返现：￥0.00元 商品总金额：¥650.00元						

图 2-14

我们试着来实现一下，先来看如下代码。

```
<html>
    <body>
        <table border="1">
            <tr>
                <th>商品编号</th>
                <th>商品名称</th>
                <th>京东价</th>
            </tr>
            <tr>
                <td>10808005</td>
                <td>史蒂夫•乔布斯传</td>
                <td>&yen;51.00</td>
            </tr>
            <tr>
```

```
            <td>529236</td>
            <td>微星主板</td>
            <td>&yen;599.00</td>
        </tr>
    </table>
</body>
</html>
```

我们看到整个表格被放到一个<table>标记里，默认的情况下，<table>的边线宽度是 0，你看不到表格线，所以要设置属性 border 为 1，<table>里面是<tr>，每个<tr>会产生一个行，<th>和<td>是单元格，其中<th>是用户表格的表头。这个表格的显示效果如图 2-15 所示。

商品编号	商品名称	京东价
10808005	史蒂夫•乔布斯传	¥51.00
529236	微星主板	¥599.00

图 2-15

其他的信息，你来补上吧。事实上京东的购物车的那个表格下面还有一行，这行不同于前三行，最后一行跨越了所有的列，代码如下。

```
<html>
    <body>
        <table border="1">
            <tr bgcolor="00CCFF">
                <th>商品编号</th>
                <th>商品名称</th>
                <th>京东价</th>
            </tr>
            <tr>
                <td>10808005</td>
                <td>史蒂夫 • 乔布斯传（Steve Jobs：A Biography 乔布斯唯一正式授
权传记简体中文版）</td>
                <td>&yen;51.00</td>
            </tr>
            <tr>
                <td>529236</td>
                <td>微星主板</td>
                <td>&yen;599.00</td>
            </tr>
            <tr>
                <td colspan="3" align="right">
                重量总计：1.25kg　原始金额：￥650.00 元 - 返现：￥0.00 元<br/>
                商品总金额：￥650.00 元
                </td>
            </tr>
        </table>
```

```
    </body>
</html>
```

为了能够看到效果，我增加了商品名称的内容，在最后一行的单元格属性中，添加了colspan="3"的内容，这样这个单元格就跨越了三列。

另外我为第一行设置了浅蓝色，有了这些基础，你能够完全完成这个表格了吧，图片还是那么设置，字体、颜色之类和过去一样。

请千万不要觉得自己已经会了，用不着彻彻底底的完成这个任务，这是作为一个程序员的态度问题，从学习就是一丝不苟，在工作中才能完成完美的作品，要知道在工作中完成再多的功能，都不如完成高质量的功能有意义。

任务中的图片不需要我多说，你完全可以从京东的网页上获得，当然更换别的商品未尝不可。实现的代码如下。

```
<html>
    <body>
        <table border="1">
            <tr bgColor="00ccff" align="center">
                <td width='7%' >商品编号</td>
                <td>商品名称</td>
                <td width='14%'>京东价</td>
                <td width='8%'>返现</td>
                <td width='8%'>赠送积分</td>
                <td width='9%'>商品数量</td>
                <td width='7%'>删除商品</td>
            </tr>
            <tr align="center">
                <td>10808005</td>
                <td><a href='http://www.360buy.com/product/10 808005.html'>
                    <img src='jobs.jpg' width="30" , height="30" border=
'none'align="top"/>
                    史蒂夫·乔布斯传（Steve Jobs：A Biography 乔布斯唯一正式授
权传记简体中文版）</a>
                </td>
                <td><font color="red">&yen;51.00</font></td>
                <td>&yen;0.00</td>
                <td>0</td>
                <td><a href=" title='减一 '><img src='bag_close. gif'
border='none' /></a>
                 <input type='text' name='add' maxlength='4' size="1"
value='1'/>
                 <a href=" title='加一 '><img src='bag_open.gif' border=
'none' /></a>
                   </td>
                <td><a href=''>删除</a></td>
            </tr>
            <tr align="center">
```

```
                <td>529236</td>
                <td><a href='http://www.360buy.com/product/529 236.html'>
                    <img src='msi.jpg' width="30" , height="30" border=
'none'align="top"/>
                    微星（msi）Z68S-G43 (G3)主板（IntelZ68 (B3)/L GA1155）
</a>
                </td>
                <td><font color="red">&yen;619.00</font></td>
                <td>&yen;0.00</td>
                <td>0</td>
                <td><a href='' title='减一 '><img src='bag_close. gif'
border='none' /></a>
                    <input type='text' name='del' size='1' value='1'/>
                      <a href='' title='加一 '><img src='bag_ open.gif'
border='none' /></a>
                </td>
                <td><a href=''>删除</a></td>
            </tr>
            <tr>
                <td colspan="7" align="right">
                重量总计：1.25kg   原始金额：&yen;670.00 元 -
返现：&yen;0.00 元<br />
                <font size="4"><b>商品总金额：</font>
                <font color="red">&yen;670.00 元</font></b></td>
            </tr>
        </table>
    </body>
</html>
```

有些地方需要说明，很小的加一和减一是使用图片来实现的，图片可自行获取。图片做了超链接会有蓝色的边框，通过设置 border="none"来消除。我们还可以设置宽度和高度来放大和缩小图片。

从表格中可以看到，我可以定义行的宽度和高度，单元格的宽度和高度，但是这里的表格会忽略完全不合理的设置，比如，如果你的设置无法显示表格中的内容，那么显示的时候并不会完全遵守设置。

其他我没有解释到的新标记和属性，请自己理解，这种能够表现出来的代码，如果不理解就去掉代码看看界面有什么变化，便可以理解了，去掉的时候为了避免难以还原，可以将其注释掉，HTML 的注释是<!-- -->。

上述代码可以说我是尽了全力，反复不断地尝试，力求接近于京东商城的购物车，但却始终达不到那样的精细程度，字体比原版的要大，将字体设置小一点并不是不可能，问题是在<table>标记外面设置<font>不起作用，因为<table>标记本身对字体有默认的设置，更小范围的设置最终会起作用，但那只能在每个单元格中设置<font>，这样就意味着要设置很多的<font>，到目前为止没有什么更好的解决办法，后面我将介绍更加强大的控制 HTML 的技术。

在使用表格时，我们通过设置 colspan 属性让一个单元格跨越多个列，你一定能猜想到还有 rowspan 属性用于跨越行。看下面的代码。

```
<html>
    <body>
        <table border="1" width="300">
            <tr>
                <td rowspan="2"> </td>
                <td> </td>
                <td> </td>
            </tr>
            <tr>
                <td> </td>
                <td> </td>
            </tr>
            <tr>
                <td> </td>
                <td> </td>
                <td> </td>
            </tr>
        </table
    </body>
</html>
```

在表格中，如果单元格里面没有任何内容，那么这个单元格不会被显示出来，所以我在每个单元格中放入 ，但是仅仅一个空格，看起来并不舒服，所以我将表格的宽度设置成 300 像素，我的电脑屏幕宽度是 1024 像素。

第一行的第一个单元格跨越两行，那么第二行就要少一个单元格，上述代码效果如图 2-16 所示。

图 2-16

我想你现在能够感受到这一组<table>标记是非常强大的，可以设置任意表格，可以任意设置表格的大小，加上我们可以将内容放到单元格中，于是就有人想到使用表格来进行定位。

2.5 用表格定位搜狐邮箱的用户登录界面

事实上，之前我们所实现的搜狐邮箱的用户登录界面还有一个大问题，就是我们是将那个大图片设置成网页的背景来使用它，但是网页的背景真的不是这幅图片，再说做成背景，将来

也没法移动，你或许能想到用<img>标记，但是用<img>并不太适合作为底图，现在如果用表格就好办多了，因为表格有相应的属性，可以将一幅图片作为表格的背景，甚至一个单元格的背景。

看如下代码效果。

```
<html>
    <body>
        <table   border="1"   background="pic_login.gif"   width="420"
height="265">
            <tr height="50">
                <td colspan="4"> </td>
            </tr>
            <tr>
                <td width="20%"> </td>
                <td colspan="2" width="45%"> </td>
                <td> </td>
            </tr>
            <tr>
                <td> </td>
                <td colspan="2"> </td>
                <td> </td>
            </tr>
            <tr height="20">
                <td> </td>
                <td width="25%"> </td>
                <td colspan="2"> </td>
            </tr>
            <tr height="50">
                <td> </td>
                <td colspan="2"> </td>
                <td> </td>
            </tr>
            <tr height="50">
                <td colspan="4"> </td>
            </tr>
        </table>
    </body>
</html>
```

在这段代码中，我综合运用了列合并，表格、行高度已经单元格宽度设定，得到如图 2-17 所示的效果，并且在这段代码中，我首次在设置宽度时运用了百分比设定。为了能够看得清楚，我将 border 属性设置为 1。

你自行对比一下用户登录界面的成品，就可以知道，我已经为每个组件打上了格子，现在我将这些组件一一对应的放到相应的格子中。

图 2-17

```
<html>
    <body>
        <form action="login.jsp">
        <table    border="1"    background="pic_login.gif"    width="420"
height="265">
            <tr height="50">
                <td colspan="4"> </td>
            </tr>
            <tr>
                <td width="20%" align="right">
                    <font size="2">邮 箱 : </font>
                </td>
                <td colspan="2" width="45%">
                    <input type="text" name="mail" size="25"/>
                </td>
                <td> </td>
            </tr>
            <tr>
                <td align="right">
                    <font size="2">密 码 : </font>
                </td>
                <td colspan="2">
                    <input type="password" name="pass" size= "27"/>
                </td>
                <td><a href=""><font size="2">忘记密码</font> </a></td>
            </tr>
            <tr height="20">
                <td> </td>
                <td width="25%">
                    <font size="2">
                    <input type="checkbox" name="save"/>记住登录状态
                </font>
                </td>
                <td colspan="2">
```

```
                <font size="2">
                <a href="">https 安全访问</a>
                </font>
            </td>
        </tr>
        <tr height="50">
            <td> </td>
            <td colspan="2"><input type="image" src="button _login.
gif"/></td>
            <td> </td>
        </tr>
        <tr height="50">
            <td colspan="4"> </td>
        </tr>
    </table>
    </form>
</body>
</html>
```

我没有去掉 border 属性，这样布局中所有的线还保留着，看完效果，去掉这个属性就是成品了，这个结果比之前我们所做的更接近原版的网页，如图 2-18 所示。

图 2-18

我再次强调自己动手不断尝试的重要性，HTML 就像是围棋，规则十分简单，但是如果想成为高手，需要大量的实战经验，这不仅仅是看得懂的问题，如果要追求细节，你需要设置大量的属性，而学习这些属性的运用才是你学习的重点，如果不细致的实现网页，很多东西可能你都不需要，你也该清楚我写这本书的原则，要么不讲，要么就实战，你完全可以去模仿更多常见的网页，HTML 写多了，经验才能有。

下面我们来看如何将这个部分放到屏幕的右边，垂直中间的位置，如图 2-19 所示。

首先获得网页上的这些图片，有些内容是 Flash，有些图片是用 JavaScript 显示到网页上的，这些没法直接另存为图片，你可以用拷屏的方式获得，如果不知道怎么做的话，到网上搜索。

图 2-19

如同登录输入的那个部分一样，对于整个屏幕，我们也用表格对每个部分进行定位，先分析一下这个网页，我们需要什么样的表格，加上 border 试着自己去规划表格。

```
<html>
    <body>
        <table border="1" width="75%" align="center">
            <tr height="125">
                <td colspan="2"> </td>
            </tr>
            <tr height="267">
                <td> </td>
                <td> </td>
            </tr>
            <tr height="150">
                <td colspan="2"> </td>
            </tr>
            <tr height="150">
                <td colspan="2"> </td>
            </tr>

        </table
    </body>
</html>
```

上述代码大体的输出了如图 2-20 所示的表格，对照一下成品，我们还要在第二行的第一个单元格里嵌套一个新的表格，下面我们就在那里再做一个表格。

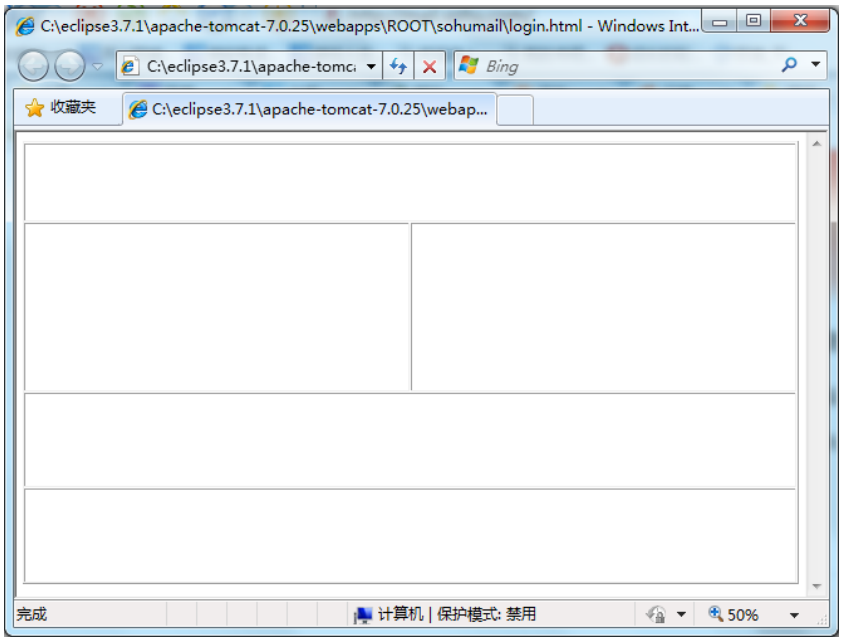


图 2-20

```
<html>
    <body>
        <table border="1" width="75%" align="center">
            <tr height="125">
                <td colspan="2"> </td>
            </tr>
            <tr height="267">
                <td width="50%">
            <table border="1" width="100%" height="100%">
                <tr>
                    <td rowspan="2"> </td>
                    <td> </td>
                    <td> </td>
                </tr>
                <tr>
                    <td colspan="2"> </td>
                </tr>
            </table>
                </td>
                <td> </td>
            </tr>
            <tr height="150">
                <td colspan="2"> </td>
            </tr>
            <tr height="150">
                <td colspan="2"> </td>
```

```
            </tr>

        </table
    </body>
</html>
```

你发现我越来越多的使用百分比来设置宽度，你可以对比使用百分比和像素点的效果，自己体会其中的区别，上述代码实现的效果如图 2-21 所示，具体到安置页面内容时，我想还是要对位置进行微调。

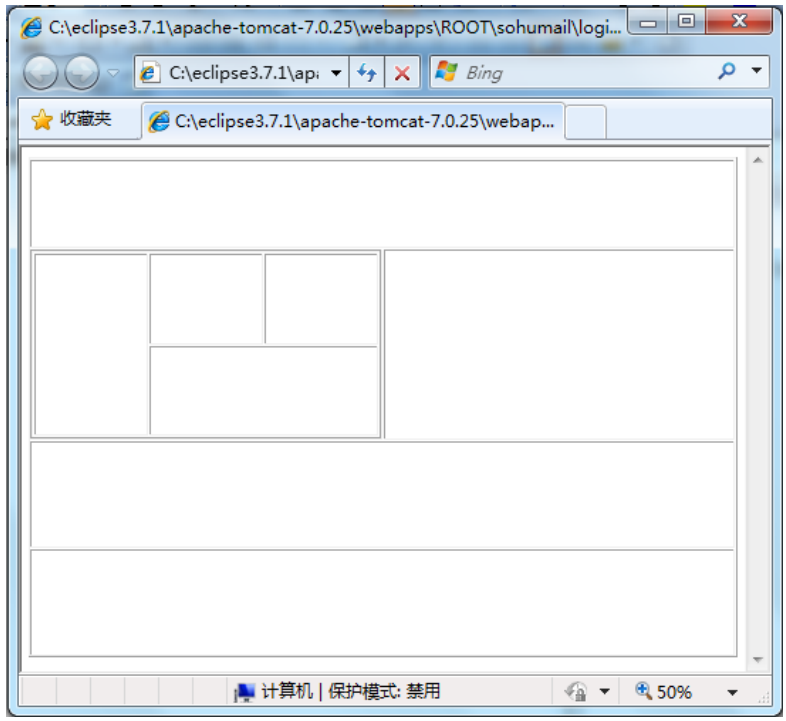


图 2-21

在这个基础上，将内容填入相应的单元格中，包括之前我们实行的登录输入部分。为了不占用太多纸张，我尽可能的少使用代码，下面的一些细节还可以更加优化，以便更加接近目标效果。同样的理由，我也不完全遵守缩进规则，但是努力让你能够容易看懂。

```
<html>
    <body>
        <table border="1" width="75%" align="center">
            <tr height="125">  <!--第一行标题图片-->
                <td colspan="2" background="title.jpg"></td>
            </tr>
            <tr height="267">
            <td width="50%"><!--第二行第一个单元格，又用了一个新表格分隔-->
            <table border="1" width="100%" height="100%">
```

```
            <tr>
                <td rowspan="2" width="30%"> </td>
                <td>  <img src="2012icon.jpg"/></td>
   <td align="center"><font size="4" color="00CCFF">祥龙贺岁</font><br/>
   <font size="3" color="AAAAAA">新的一年开启新的<br/>希望，祝您新年快
<br/>
乐，吉星高照！</font></td>
            </tr>
            <tr>
                <td colspan="2">
          <!--无序列表，相关内容自行上网查找，使用的不太多-->
          <ol>
              <li><span>闪电般的速度</span>页面响应迅速</li>
              <li><span>安全稳定的服务</span>国内一流服务品质</li>
              <li><span>50M 超大附件</span>目前国内最高的附件大小</li>
          </ol>
                </td>
            </tr>
          </table>
          </td>
          <td>
          <!--放入之前实现的用户登录代码-->
       <form action="login.jsp">
       <table   border="0"   background="pic_login.gif"   width="420"
height="265">
          <tr height="50">
              <td colspan="4"> </td>
          </tr>
          <tr>
              <td width="20%" align="right">
                  <font size="2">邮 箱 ： </font>
              </td>
              <td colspan="2" width="45%">
                  <input type="text" name="mail" size="25"/>
              </td>
              <td> </td>
          </tr>
          <tr>
              <td align="right">
                  <font size="2">密 码 ： </font>
              </td>
              <td colspan="2">
                  <input type="password" name="pass" size= "27"/>
              </td>
              <td><a href=""><font size="2">忘记密码</font></a> </td>
          </tr>
          <tr height="20">
              <td> </td>
```

```
                <td width="25%">
                    <font size="2">
                    <input type="checkbox" name="save"/>记住登录状态
                    </font>
                </td>
                <td colspan="2">
                    <font size="2">
                    <a href="">https 安全访问</a>
                    </font>
                </td>
            </tr>
            <tr height="50">
                <td> </td>
                <td colspan="2"><input type="image" src="but ton_login.
gif"/></td>
                <td> </td>
            </tr>
            <tr height="50">
                <td colspan="4"> </td>
            </tr>
        </table>
        </form>
            </td>
            </tr>
            <tr height="150">
            <!--第三行，广告-->
                <td colspan="2" align="center"><img src="ad. jpg"/></td>
            </tr>
            <tr height="150">
            <!--第四行，最下面的版权信息等-->
                <td colspan="2" align="center">
    Copyright &copy; 2012 Sohu.com Inc. All rights reserved. 搜狐公司 版
权所有<br/>
    京 ICP 证 000008 号 北京市通信公司提供网络带宽<br/>
    <img src="net110.gif"/>
    <a href="mailto:webmaster@vip.sohu.com">mailto:webmaster@vip.sohu.
com</a>
    <br/>热线电话： 010-58511234 
    <a href="http://me.sohu.com/online_service.jsp?pid=9">在线客服
    <img src="user_on_1_0000.gif" ></a>
    <img height=50 src="pic21.gif" width=40 >
    </td>
    </tr>
    </table>
    </body>
</html>
```

去掉 border 属性后，我们得到的网页效果如图 2-22 所示。

图 2-22

看样子已经基本达到了目标效果了，你该停下来到网上找些属性的网页，自己来实现一下，借此不断地练习，获得实战经验，不知道的标记和属性，都可以在网上搜索到，HTML 现在应用如此广泛，网上的资料非常多。

按说掌握到这，你已经能够胜任完成网页制作的任务了，传统的 HTML 课程只是比我所涉及的更加详细而已，范围上并没有更多的东西。

但是想必你也发现了，我们现在制作网页的过程相当痛苦，简单地讲，我们就是不断地在页面上做表格、设置表格，在单元格中再做表格，这样整个页面都是被表格分隔着的，对于更加复杂的页面，表格也是十分复杂的，而且这样的网页也遗留了维护问题，想象一下，如果我们要将用户输入的那个模块移动到右上角，这是一件很困难的事情，可能要修改整个表格的系统。

在经历了很长的一段时间后，HTML 技术也有所发展，提供了新的属性，允许进行直接定位，而这个发展远不是定位这么简单，它甚至替代了 HTML 的大部分标记功能，并且在使用上更加强大，这样革命性的发展不再是 HTML 的一部分，而是独立出来的一个新的语言，叫做 CSS，CSS 需要和 HTML 配合使用，所以下一步我们将使用 HTML+CSS 来实现网页。事实上搜狐邮箱的用户登录页面本身就是用 HTML+CSS 实现的。

2.6　使用 CSS 实现搜狐邮箱的用户登录

2.6.1　绝对定位

随书的光盘上提供了 CSS 的参考手册，我们先用最简单的一句话来尝试一下自由定位，找到 CSS 的参考手册，双击运行后，能看到如图 2-23 所示的界面。

图 2-23

单击界面左栏的“属性”选项，我们主要查找的内容就是 CSS 的这些属性，在“属性”选项中单击“定位”选项，我们看到第一个属性是 position，选择后，右边部分介绍了这个属性的使用方法，我们看到 position 属性有四个可能的值，static 是默认值，说明如果设置了这个值，这个属性设置就算白写。仔细看 absolute，这就是我们需要的值，如果设置为 absolute，设置的内容将不再遵守文档流的顺序，进一步介绍要使用 left、right、top 和 bottom 进行绝对定位，这四个属性一看就知道分别对应着左、右、上、下。

想必这四个属性的设置大同小异，我们找来其中一个看看，比如 top，抛开默认值，就是设置长度了，这正是我所期望的，其中的解释告诉我可以用长度值或是百分数，说明这里也可以使用百分比，在文档下面的例子中，我们看到了长度值后面还有长度单位，而说明后面是长度单位的超链接，单击进去看看，没想到有这么多的长度单位，这其中 px 是我使用的比较多的，px 对应的是像素点，因为我一直清楚我的屏幕是 1024×768 的分辨率，也就是说这个屏幕上有 1024×768 个像素点，所以使用 px 为长度单位相对比较容易理解设置的长度是多少，当然每个人的习惯不同，不排除你会相对比较喜欢其他的单位，但是我还是要推荐相对长度单位，因为我们不知道用户电脑设置的分辨率是多少，相对长度单位可以随着用户的设置适当调整。

下面我将一句话设定到页面的一个位置上去。

```
<html>
    <body>
        <a style="position:absolute;top:100px;left:100px;">测试 CSS</a>
    </body>
</html>
```

2.6.2　div

我们查询到的属性被放到 HTML 标记的属性 style 中，并且遵循着一个语法格式：CSS 属性:值;。在一个 style 中可以设置多个 CSS 属性。在这个例子中，我使用了 HTML 中的标记<a>，事实上你可以在 HTML 中任何标记里面使用 style，通过 style 设置 CSS 属性，因为 style 作为 HTML 属性不能单独存在，所以我们借用了标记<a>，问题是标记<a>有自己的功能，其实在使用了 CSS 后，这些 HTML 标记的功能还存在，除非被 CSS 的相同功能覆盖。

你完全可以将<a>变成<b>看看会发生什么变化，也可以修改一下 top 或 left 的值，改变长度单位，看看会发生什么变化。

查询一下 CSS 文档，看看能不能找到设置粗体的属性，我找到了一个属性是 font-weight，normal 的值是设置字体为正常值，bold 用来设置加粗，那么，如果标记是<b>，而 CSS 设置成正常值会是什么样子呢？

```
<html>
    <body>
    <b style="position:absolute;top:100px;left:100px;font-weight:normal;">
测试 CSS</b>
    </body>
</html>
```

你能看到结果，最终效果依照 CSS 的设定，标记的效果被取代了，这就带来了一个问题，CSS 如此强大，标记<b>是否有存在的意义，答案是有的，之前我说大部分 HTML 标记都将被 CSS 属性所替代，替代方式就是使用标记<b>，这要比使用 CSS 代码更简洁，因此在大多数情况下，我们能够看到 HTML 标记和 CSS 是混合使用的，CSS 设置的遵循 CSS，CSS 没有设置的遵循 HTML 标记。

在使用中，人们发现某一标记一旦使用了 CSS，HTML 标记的简洁性也就无从谈起了，因为 CSS 是集中设置的，而 HTML 标记原本的设置功能有时反倒成了累赘，要是有一个标记，什么功能都没有就好了，这个需求在 CSS 出现之前是不可能实现的，现在这个什么功能都没有的标记 反倒成了 HTML 中非常重要的标记。

使用 上述的代码可以这样描写。

```
<html>
    <body>
    <div style="position:absolute; top:100px; left:100px;">测试 CSS</b>
    </body>
</html>
```

理论上 +CSS 可以替代大多数 HTML 标记，但是实际工作中也有些 HTML 标记 CSS 认为没有必要替代，比如超链接，而有些标记虽然 CSS 中有替代属性，但是两者各有千秋，比如进行复杂的局部定位可能使用<table>会更简洁。

2.6.3 级联样式

其实 CSS 并不是仅仅用于替代 HTML 标记的，我们之前遗留了一个问题一直没法解决，用于输入用户名的 text 文本框和 password 文本框没法对齐，原因我们也知道，因为这两个文本框的字体是不同的，但是我们又无法设置其中的字体，因为我们的设置会被<input>标记的默认设置替代，现在有了 style，我们就可以进行字体设置了。建议你先通过自己查询 CSS 文档，尝试这样做，然后再看代码。

```
<html>
    <body>
        <form action="">
            用户名:
        <input type="text" name="user" style="font-family:宋体"/><br/>
            密  码:
        <input  type="password"  name="pass"  style="font-family: 宋 体
"/><br/>
            <input type="submit" value="登录"/>
            <input type="reset" value="重新输入"/>
        </form>
    </body>
</html>
```

这样设置后，文本框终于对齐了，在我们这个案例中有个问题并不明显，如果是用户注册，那么需要的文本框将会很多，加上页面设置时往往会有更多要求，比如除了字体，还可能设置颜色，大小之类。每个文本框都这样设置未免太啰嗦了，CSS 提供了统一进行设置的能力。

到 CSS 文档中选择“简介”中的“样式表简介”选项，我们能看到其中有“如何将样式表加入您的网页”选项，其中介绍了三个加入样式表的途径，我们之前使用的无疑是“内联定义”，下面我们来看“定义内部样式块对象”的做法。

我们发现 CSS 的语法被放到了<style>标记中，令人费解的是具体的 CSS 语句竟然被注释掉了，这是 HTML 语法发展过程中的一个问题，我们设想一下，现在大家都在用 HTML1，那么所有的浏览器自然是只支持 HTML1 的，所有的网页也是符合 HTML1 标准的，现在 HTML2 出现了，在 HTML2 中必定会加入一些新的标记，问题是世界上大多数的浏览器并不认这些新标记，作为网页编写者，会在获得更好的页面效果和让更多人能够看到自己的网页之间进行权衡，通常让更多人能够看到自己的网页的想法会占上风，这意味着没法使用新的标记，那么如果互联网上所有的网页都只使用 HTML1，用户也就没有必要升级浏览器了，HTML 的发展因此被这样的局面束缚住，直到有人想出了一个办法，将新的标记放到注释中，这样旧的浏览器会自动忽略这些新标记，而新的浏览器在设计时会去看注释中的内容，这样网页就能即兼顾旧的浏览器，也能使用新标记了，我们在很多网页中能够看到这样的现象，将代码写到注释中，只是在 CSS 技术的这个部分，现在不再需要这样做了，今天的浏览器已经足够的新了。

现在依照此做法再重新实现上面的代码，具体实现代码如下。

```
<html>
    <head>
        <style type="text/css">
input{
    font-family:宋体;
    font-size:18px;
}
        </style>
    </head>
    <body>
        <form action="">
            用户名: <input type="text" name="user"/><br/>
            密  码: <input type="password" name="pass"/> <br/>
            <input type="submit" value="登录"/>
            <input type="reset" value="重新输入"/>
        </form>
    </body>
</html>
```

我们看到使用定义方法，语法有了一些改变，花括号前面是 HTML 的标记名称，花括号里面还是原本的 CSS 语法格式，只不过因为写了出来，所以可以看起来更舒服些。我特地将字体设置得大了些，这样能够明显地看出效果的不同，而设置字体大小的时候，我们也能发现，但就这个方面 CSS 也比纯粹的 HTML 标记能力要强得多，你可以非常精确地设置字体大小。

最主要的是这样的 CSS 设置，一下子影响到了所有的<input>标记，因此才有了 CSS（级联样式表）这个名字，虽然大多数情况下我希望<input>的组件样式一致，但是也有可能在某些任务中不希望这样。我可以用不同的选择符。

比如下面的代码。

```
<html>
    <head>
        <style type="text/css">
#input{
    font-family:宋体;
    font-size:18px;
}
#button{
    font-family:黑体;
    font-size:12px;
}
        </style>
    </head>
    <body>
        <form action="">
            用户名: <input type="text" name="user" id="input"/> <br/>
            密    码 : <input type="password" name="pass"
id="input"/><br/>
            <input type="submit" value="登录" id="button"/>
```

```
            <input type="reset" value="重新输入" id="button"/>
        </form>
    </body>
</html>
```

再来看最后一个我们没有涉及到的加入样式表的办法“链入外部样式表文件”，我们需要在.html 文件以外，再创建一个扩展名为.cs 的文件，在这个例子中，我创建的 CS 文件为 login.cs，在 CS 文件中写入 CSS 代码。

```
#input{
    font-family:宋体;
    font-size:18px;
}
#button{
    font-family:黑体;
    font-size:12px;
}
```

然后我们依照文档将这个 CS 文件链接进来，具体代码如下。

```
<html>
    <head>
    <link rel="stylesheet" href="login.cs" type="text/css"/>
    </head>
    <body>
        <form action="">
            用户名：<input type="text" name="user" id="input"/> <br/>
            密    码 ： <input type="password" name="pass" id=
"input"/><br/>
            <input type="submit" value="登录" id="button"/>
            <input type="reset" value="重新输入" id="button"/>
        </form>
    </body>
</html>
```

你完全可以将 CS 文件中的属性变换一下，这样测试的时候就能看出变换的效果了。

我在 href 属性中链接 CS 文件时，做法和文档是有所不同的，为了简化学习过程，我将 CS 文件和 HTML 文件放到了一个目录中，这样我们几乎不用关心路径问题，而文档中举例时用的是基于互联网的 URL 格式，在工作中或许这样的做法更加普遍，这里有一个深层度的理由。

我们来看将 CSS 语法分离出去，成为一个独立的文件，所带来的革命性变化。

在这之前，HTML 和 CSS 是写在一起的，这样在一个文件中不可避免的混杂着两种不同的语法，现在看来这也不是什么大不了的问题，学两个语法就是了，但是要知道在软件企业里，专业化分工有越来越明确的趋势，一个专心 5 年的 HTML 程序员或是一个专心 5 年的 Java 程序员，与 5 年什么技术都涉足的程序员相比，编出的代码质量将会有很大的不同，我们在学习阶段要学习很多技术来提高在这个行业中的适应能力，但是终将有一天，我们会成为某个领域的专家。将 HTML 和 CSS 分离在两个不同文件的做法，支持了技术人员的分工。

彻底分离的结果是 HTML 文件专门提供网页上的内容，另一个 CS 文件进行页面的设计，而不是将内容和页面设计混合在一起，这样提供内容的工作就完全可以用程序来完成，这也为 Java Web 编程奠定了坚实的基础。

我们甚至可以在一个网页背后，准备多个 CS 文件，这样网页会拥有不同的版本，使用时只需要链接不同的 CS 文件即可，想象一下，如果我们的网站需要给 PC 客户端看，还需要给手机，以及平板电脑看，虽然在这些设备上显示的内容是一致的，但是由于分辨率的不同，要求网页拥有不同的样式，那么提供一个内容文件，不同的用户链接不同的 CS 文件，这是最合理的解决方案。

下面再来实现一遍搜狐邮箱的用户登录界面，先写出提供内容的 HTML 文件。将 HTML 文件命名为 maillogin.html，而 CS 文件我将其命名为 maillogin.cs。

你务必依照之前讲解练习的内容，加上 CSS 文档，先自己尝试，然后再看我的代码。下面是 HTML 代码。

```
<html>
<head>
<link rel="stylesheet" href="maillogin.cs" type="text/css"/>
<title>搜狐闪电邮箱——十年专业品质，国内一流的免费邮箱</title>
</head>

<body>
<div id="bg">
    <div id="top"><img src="title.jpg"/></div>

    <div id="icon"><img src="2012icon.jpg"/></div>

    <div id="longtitle">祥龙贺岁</div>

    <div id="mess">新的一年开启新的希望，祝您新年快乐，吉星高照！</div>

    <div class="left">
        <ol>
        <li><span>闪电般的速度</span> 页面响应迅速</li>
        <li><span>安全稳定的服务</span> 国内一流服务品质</li>
        <li><span>50M 超大附件</span> 目前国内最高的附件大小</li>
        </ol>
    </div>

    <div class="bground"><img src="pic_login.gif"/></div>

    <div class="right" id="login_box">
    <form action="">
    <ul>
        <li>邮 箱: <input type="text" name="user" tabindex="1"/></li>
             <li></li>
```

```
        <li>密 码：<input type="password" name="pass" tabindex="1"/>
             <a href="" class="pw">忘记密码</a></li>
        <li></li>
        <li  class="mar"><input  name="cookie"  type="checkbox"  value=
"1" tabindex="1"/>
             <label for="rpwd">记住登录状态</label>
             <a href="" class="pw">https 安全访问</a></li>
        <li></li>
        <li   class="mar"><input   type="image"   src="button_login.gif"
border="0"/></li>
    </ul>
    </form>
</div>

<div class="ad"><img src="ad.jpg"/></div>

<div id="bottom">
<div>Copyright &copy; 2012 Sohu.com Inc. All rights reserved. 搜狐公司
版权所有</div>
<div>京 ICP 证 000008 号 北京市通信公司提供网络带宽</div>
<div                                                  style="float:left"><a
href="http://net.china.cn/chinese/index.htm"><img src="net110.gif">
</a></div>
<div style="float:left"><a href="mailto:webmaster@vip.sohu.com">
   mailto:webmaster@vip.sohu.com</a>
   <br/>热线电话：  010-58511234 
   <a  href="">在线客服<img  style="MARGIN-LEFT:  2px"  src="user_on_1_
0000.gif"></a>
</div>
<img height=50 src="pic21.gif" width=40 ></div>
</div>

</div>
</body>
</html>
```

你发现上面的代码完全是用 将所有的内容列举出来的，如果去掉链入外部样式表的话，用浏览器看这个页面，这些内容是从上到下按顺序平铺下来。

接下来我们来编写 CS 文件，这个过程并不容易，要一边改一边看效果，好在功能确实非常强大，定位的时候可以一个像素点一个像素点的调整，所以我们完全可以实现和真正的搜狐网站一样的效果。

在 HTML 和 CSS 中，我不但使用了属性 id，还使用了属性 class，遇到 id，CSS 用#选择，class 用.选择。

```
#top{text-align:center;}

#icon{position:absolute; top:128px; left:300px;}
```

```
#longtitle{position:absolute; top:128px; left:430px; font-size:18px;
    font-weight:bold; line-height:28px; }

#mess{position:absolute; top:160px; left:410px; color:#666666;
    line-height:24px; font-size:14px; margin:0 0 30px 22px; width:118px; }

.bground{position:absolute; top:126px; left:630px;}

.left{position:absolute; top:280px; left:260px;}
.right{position:absolute; top:190px; left:670px;}

.mar{margin:-3px 0 -3px 22px;}

.ad{     position:absolute; top:400px; left:290px;}

#bottom{ position:absolute; top:500px; left:490px; text-align:center;
color:#666666;}

span{font-size:14px; font-weight:bold;}

ul,ol,li,form{ border: 0; z-index:inherit; list-style:none;}

img,a img{border:0; margin:0; padding:0;}

input{font:14px verdana, sans-serif; font-weight:bold}

/* 默认字体，颜色等 */
html,body {
    margin: 0px; padding:0; font:12px/1.6 Arial, Helvetica, sans-
serif; color:#437ca0; }

/* 链接颜色 */
a:link,a:visited{color:#437ca0; text-decoration:none}
a:hover {text-decoration:underline;}
```

CSS 的学习不是一蹴而就的，一方面涉及的内容很多，一方面 HTML 和 CSS 都非常讲求实战，所以建议你再找一些熟悉的网页，照葫芦画瓢的实现出来，锻炼着上网搜索和查询文档，如果有些效果实在实现不出来，也可以参看原版网页的源代码。

2.7　在网页上显示时间

2.7.1　为什么要学习 JavaScript

HTML 学习到现在，它和 Java 有什么不同吗？这个问题真的没法回答，因为有太多的不同

了，从根本上讲 HTML 的功能就是表达，Java 更多的是拥有行为，事实上 Java 也能表达，但是在表达能力上似乎不如 HTML 强大，那么 HTML 不需要动作吗？一定是需要的，我们在网络上也能看到很多 HTML 的动作，比如，鼠标放到那个地方，就会弹出来一个新的东西，比如，在表单输入的时候，如果你没输入用户名，会提示你用户名不能为空。

其实网页中的这些动作，并不是 HTML 实现的，HTML 没有这样的能力，是另外一个叫做 JavaScript 的语言完成的这些动作，但 JavaScript 也没有 Java 这么强大，比如，JavaScript 不能访问数据库，如果只是需要一个展示内容的网站，那么 HTML 就足够了，但是这样的网站现在是非常少的，如果希望页面能够响应用户的输入，那么就需要 Java 在后台配合，理论上讲 HTML 和 Java 一起使用，就能完成大部分事情，但是这样做的效率太低了，因为 Java 只能运行在服务器端，用户的所有输入都需要通过网络发送到服务器去，并不是计算机或网络觉得这样太辛苦了，而在这样的传送可能会让用户等待一段时间，用户也许因此失去耐心，我们说这样的网站不能实时响应。

JavaScript 的存在解决了这个问题，因为 Javascript 运行在客户的浏览器中，所以对于用户的输入可以实时响应。现在越来越流行 HTML+JavaScript+Java 的结构，HTML 用于表现内容，JavaScript 用于简单的用户响应，而 Java 则去处理复杂的逻辑。

JavaScript 是一门完全新的编程语言，它和 Java 之间没有任何关系，只是 Java 这个名字太火了，所以 JavaScript 借用了这个名字，目前人才市场上一个优秀的 JavaScript 程序员的价值远远超过了 Java 程序员。有时，我们将 JavaScript 编程称为前端编程，而 JavaScript 程序员就被称为前端程序员。

我们首先要了解一下 JavaScript 的生存环境和 Java 是不同的，Java 和很多计算机语言一样，运行在计算机的操作系统之上，它能够做任何计算机应用程序能做的事情，而 JavaScript 被嵌入到了 HTML 中，所以 JavaScript 又被称为脚本程序，思考一下脚本是个什么概念，Script 便是脚本的意思。JavaScript 程序运行在浏览器里面，浏览器运行在操作系统上，也就是说 JavaScript 能够使用的资源取决于浏览器拥有多少资源。JavaScript 也是被浏览器解释执行的，要知道很多浏览器被装入了不同的插件，这些插件提供了不同的 JavaScript 资源，加上浏览器的宽松性，JavaScript 显得比 Java 要随意多了，初学的时候，你会喜欢这样的随意性，仿佛卸下了严格语法的枷锁，但是在实际运用的时候你就会发现，严格的语法更容易驾驭。

和 CSS 一样，我在光盘中提供了 JavaScript 的帮助文档，虽然是英文的，但是因为很快你就能够写程序测试这些内容，所以很多时候，你不需要看懂其中的解释，写代码看看效果便明白了，希望你能够养成看文档，以及用代码测试效果的习惯。

2.7.2 获取时间

好吧，我们现在开始学习这门新的编程语言——JavaScript。我们先来看如何编写并运行 JavaScript 程序。前面提到过 JavaScript 是嵌入到 HTML 中的编程语言，下面的语句让 JavaScript 得以运行。别忘了，现在的任务是显示时间，所以我在下面的例子中，提供了用于显示的语句。

具体代码如下。

```
<html>
    <head>
    <script language ="JavaScript">
            alert("这是第一个 JavaScript 例子!");
        </script>
    </head>
</html>
```

JavaScript 代码被放在<script>标记中，在我所提供的例子中，还附带了一个 language 属性，说明一下代码是 JavaScript 语言，事实上这个属性可以不提供，因为默认就是 JavaScript，但是这个属性的存在说明，还有类似于 JavaScript 的语言存在，那就是 VBScript，这是微软提出的脚本语言，只是由于微软对技术的开发态度最终使得 VBScript 败下阵来，现在已经很少有人提起了。

alert 语句的运行效果，你自己试一下就知道了，alert 是个典型的方法，它会弹出一个警告对话框，里面显示参数中的那段话，因为 JavaScript 是脚本语言，是由浏览器解释执行的，所以这个语言没有编译过程，那么也就没有编译时对语法错误的检查，所以我们常常在编程的过程中，放置 alert 来确定程序在什么地方出了问题。现在正式上线的网页，已经很少使用 alert 了，因为人们越来越倾向于使用不这么干扰用户的提示方式。

JavaScript 毕竟是有一定动作能力的编程语言，本机上的 JavaScript 程序可能会是恶意的，所以如果你选择用浏览器打开本机包含 JavaScript 代码的网页时，浏览器上面会出现安全提示，只有用鼠标单击允许运行，JavaScript 的代码才能够运行起来，如果是通过 URL 进行访问，比如，通过 IP 地址或 Tomcat 访问，浏览器就会启动相应的安全策略，确保不被恶意操作，这样就没有安全提示了。

事实上 JavaScript 也是面向对象的语言，虽然大多数人不会用 JavaScript 创建类，但是至少这里有类，也有创建对象的过程。在我们的任务中，需要用到一个重要的类 Date。请看如下代码。

```
<html>
    <head>
        <script>
            d = new Date();
             alert(d);
        </script>
    </head>
</html>
```

现在看到了久违的 new，这就是创建对象，可是这和我们熟悉的 Date d = new Date();语句有一点不同，前面的类型 Date 没有了，这是因为 JavaScript 是个弱类型的语言，其实它是有类型的，只是你不需要声明，一个变量的类型取决于第一次赋值的类型，如果变量不声明让你觉得不舒服的话，可以这样写：var d = new Date() ;。假如你想看 d 的类型是什么，可以加上一条语句 alert(typeof(d)) ;，然后看一下结果，你发现即便这样 JavaScript 也比想象的弱，在这个例子中 d 的类型是 object，而不是 Date，它只能识别到 object 这个程度了。

要注意的是，JavaScript 也需要区分大小写，但是写错了没有提示，只是在显示的时候，浏览器的左下角有个叹号的错误提示，如果双击那个地方，就会出现错误信息，只是这个错误信息往往不知所云。

下一句不难理解，将变量 d 显示出来。现在我们用浏览器打开这个网页看看，如图 2-24 所示。

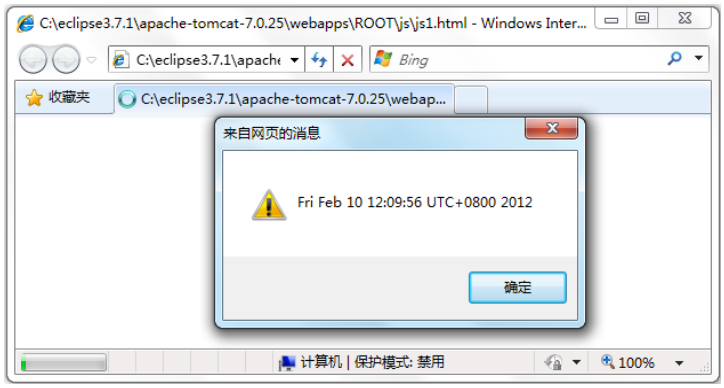


图 2-24

仔细看看，真的是时间，而且是很详细的时间显示，有年月日、星期、时分秒，甚至还显示了时区。但是这个显示似乎和我们的习惯不同，下面我将显示的时间改成习惯的格式。

既然 d 是对象的引用，那么自然会有成员方法，没错，d.getYear();能够得到年，你可以做实验来验证一下，其他的方法大体能够猜出来，我们就用它的成员方法和字符串拼接来组合新的显示格式。

```
<html>
   <head>
      <script>
         d = new Date();
         //年月日
         time = d.getYear()+"年";
         time += d.getMonth()+"月";
         time += d.getDate()+"日";

         //星期几
         time += "    星期"+d.getDay()+"    ";

         //时分秒
         time += d.getHours()+":";
         time += d.getMinutes()+":";
         time += d.getSeconds();

         //显示
          alert(time);
```

```
        </script>
    </head>
</html>
```

上面的代码不是一下子写成的，而是向字符串中添加了一组内容，下面我们来运行一下，看看效果是否和想象的那样。整个代码完成后，效果如图 2-25 所示。

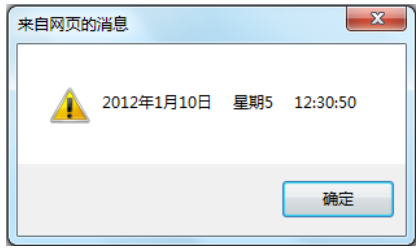


图 2-25

我们发现这其中有几个问题。一是月份不对，比当前的月份少了一个月，这是因为其实只有我们才说 1、2、3 月，在英语，月份不是数字，每个月都有名字，硬要说成数字的话，是从 0 开始数的，这个问题解决起来太简单了，遇到月份值，加上 1 再操作就行了。要说星期也有这个问题，巧的是国外的习惯是将星期日作为一个星期的第一天，这样我们的一、二、三，恰好和取的值吻合。

第二个问题还是星期，我们不习惯星期用阿拉伯数字，而且可以想象，这样的话，会显示出星期 0 的结果，这也不符合习惯，我们可以很直接地用 if 语句将阿拉伯数字转换成想要汉字，还有另一个办法，我们定义一个数组，数组里按照顺序存放日、一、二、三等结果，当我们得到星期几的数字后，就用这个数字作为数组的下标来取数组中的内容。在 JavaScript 中数组也是类，使用数组要用 new 对象。

时间上还有问题，上面的显示效果是我运气好，有时我们会遇到个位的数字，这样便看着不舒服了，如图 2-26 所示。

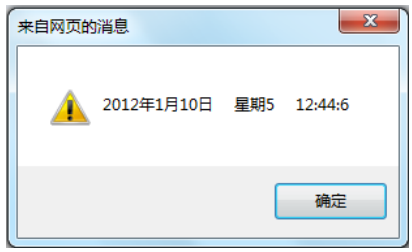


图 2-26

看到秒那个位置上的数，我们希望在这样的情况下应该显示 06，时分秒都会有这样的问题，看来只能使用 if 语句了。建议你自己先尝试，而且是写一点测试一点，在这个过程中体会这门语言。修改成习惯的时间显示格式的代码如下所示。

```
<html>
    <head>
        <script>
            dd = new Array("日","一","二","三","四","五","六");
            d = new Date();
            //年月日
            time = d.getYear()+"年";
            time += (d.getMonth()+1)+"月";
            time += d.getDate()+"日";

            //星期几
            time += "    星期"+dd[d.getDay()]+"    ";

            //时分秒
            if(d.getHours()<10) {
                time+="0";
            }
            time += d.getHours()+":";
            if(d.getMinutes()<10) {
                time+="0";
            }
            time += d.getMinutes()+":";
            if(d.getSeconds()<10) {
                time+="0";
            }
            time += d.getSeconds();

            //显示
             alert(time);
        </script>
    </head>
</html>
```

显示效果如图 2-27 所示。

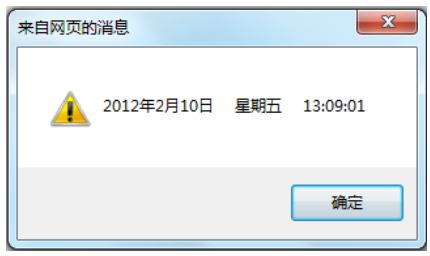


图 2-27

2.7.3 定义函数

你发现为了确保时分秒都是两位显示，我用了逻辑上完全一样的 if 语句，在这种情况下，

我们会想到定义一个函数来实现这个逻辑。

```
<html>
    <head>
        <script>
            dd = new Array("日","一","二","三","四","五","六");
            d = new Date();
            //年月日
            time = d.getYear()+"年";
            time += (d.getMonth()+1)+"月";
            time += d.getDate()+"日";

            //星期几
            time += "    星期"+dd[d.getDay()]+"    ";

            //时分秒
            time += addzero(d.getHours())+":";
            time += addzero(d.getMinutes())+":";
            time += addzero(d.getSeconds());

            //显示
             alert(time);

            function addzero(i) {
                if(i<10) {
                    return "0"+i;
            }
                return i;
            }
        </script>
    </head>
</html>
```

函数的定义可能是迄今为止和 Java 语言差别最大的语法了，我们要写 function 关键字，说明这是个函数，由于在 JavaScript 中变量不需要声明类型，所以函数也不需要声明返回值类型，参数也就不需要声明类型了。

声明函数 addzero 时，我们发现原本写的程序不属于任何函数，甚至没有 main 函数存在，这也是脚本语言的重要特征。

2.7.4　js 文件

几乎和 CSS 的理由一样，将 JavaScript 代码分离出去，成为一个独立的文件，可以使网页的内容和网页上面的动作分离开。分离出来的文件扩展名是 js，所以 JavaScript 程序员常常说自己是写 js 的。

分离出 js 文件的理由要比分离 cs 文件的理由充分得多，我们一直编写的显示时间的代码估

计在很多地方都会用到，每次都写一遍未免太麻烦，现在我们可以将其写成一个 js 文件，以后只要网页中要显示时间，只需调用进来就可以了。

但是直接将写的指令分离出去是不起作用的，我们将这些逻辑都包装到函数中，然后在 HTML 里面调用就是了。

js 文件中的代码如下。

```
function showTime() {
    dd = new Array("日","一","二","三","四","五","六");
    d = new Date();
    //年月日
    time = d.getYear()+"年";
    time += (d.getMonth()+1)+"月";
    time += d.getDate()+"日";

    //星期几
    time += "    星期"+dd[d.getDay()]+"    ";

    //时分秒
    time += addzero(d.getHours())+":";
    time += addzero(d.getMinutes())+":";
    time += addzero(d.getSeconds());

    //显示
    window.status = time;

    //隔一秒种调用一下自身
    setTimeout("showTime()" , 1000);
}

function addzero(i) {
    if(i<10) {
        return "0"+i;
    }
    return i;
}
```

HTML 文件调用 js 文件的代码如下，假设 js 文件的名字是 showTime.js。

```
<html>
    <head>
        <script language="javascript" src="showTime.js"></script>
        <script>
            showTime();
        </script>
    </head>
</html>
```

需要强调的是，引入外部 js 文件的<script>标记并不包含什么内容，但是不可以在标记末尾

使用斜线结束，<script language="javascript" src="showTime.js"/>无法成功引入 js 文件。

这样我们就不再需要每次都重新输入时间格式的转换代码了，但是在实际工作过程中，我发现有些 JavaScript 语句不能分离出去，如果你遇到 JavaScript 程序没有反应，试着写到一起来解决问题。

2.7.5　显示到其他地方

用 alert 显示时间并不舒服，因为 alert 会中断用户的操作，时间也不是重要到必须中断用户操作的信息，现在我们来学习如何将时间显示到其他地方。

首先我们来看如何将信息显示到浏览器下面的状态栏中，我们知道 JavaScript 是运行在浏览器中的，那么有些浏览器资源就天然的成为 JavaScript 的资源，作为编程语言，资源被表现成对象的形式，特别的是这些对象不需要由程序员创建，因为他们天然的就存在，比如浏览器本身，对于 JavaScript 来说，就是一个天然存在的对象，我们称其为内置对象，浏览器在 JavaScript 中的对象引用的是 window，因为这个对象太显而易见了，所以很多时候我们将 window 省略掉，其实 alert 就是 window 对象的方法，在之前的代码中我们便是省略了 window。

在 window 这个对象里面还包含其他的方法和属性，方法的含义和 Java 中相似，但属性和 Java 的成员变量并不相同，后面我们会深入接触属性，现在可以先理解成为成员变量，浏览器下边的状态栏便是其中的一个属性 status。

window.status = "浏览器状态栏显示";

测试完这条语句，你试着将时间显示到状态栏上。效果如图 2-28 所示。

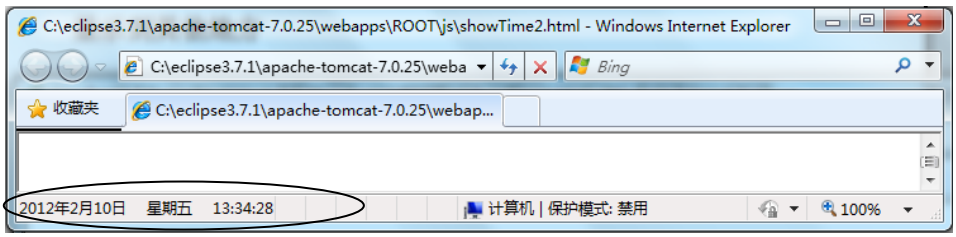


图 2-28

那么能不能显示在网页的页面上呢？或许你能够猜到，状态栏是 status，那么页面必定也是一个属性，没错，浏览器中显示网页的那个区域叫做 document，document 也是个对象，是包含在 window 中的对象。

document 有一个方法 write，能够将一些信息写到页面中，写的信息支持 HTML 的语法。

window.document.write("显示在网页中") ;

现在你试着将时间显示在网页中吧。如图 2-29 所示。

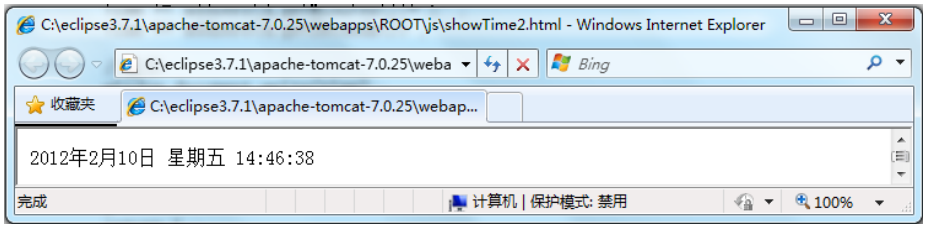


图 2-29

2.7.6 能动的时间

让时间真正动起来是个自然而然的需求，学习 Java 的过程中，了解到产生动画的效果需要线程，因为线程能够通过代码，产生隔一段时间调用一次的效果，JavaScript 中没有线程概念，但是也能隔段时间调用一次。

既然说到调用，自然是针对函数的调用了，让时间走起来，就意味着每秒钟新建一个 Date 对象，这样才能取得新的时间，并经过格式转换后将其显示出来，所以我们还是要将代码包装到一个函数中。例如如下代码。

```
<html>
    <head>
        <script>
        //第一次要调用一下
        showTime();

        //显示时间的函数
        function showTime() {
            dd = new Array("日","一","二","三","四","五","六");
            d = new Date();
            //年月日
            time = d.getYear()+"年";
            time += (d.getMonth()+1)+"月";
            time += d.getDate()+"日";

            //星期几
            time += "    星期"+dd[d.getDay()]+"    ";

            //时分秒
            time += addzero(d.getHours())+":";
            time += addzero(d.getMinutes())+":";
            time += addzero(d.getSeconds());

            //显示
            window.status = time;

            //隔一秒种调用一下自身
```

```
            setTimeout("showTime()" , 1000);
        }

        //确保显示两位数字
        function addzero(i) {
            if(i<10) {
                return "0"+i;
            }
            return i;
        }
        </script>
    </head>
</html>
```

上述代码在浏览器的状态栏中会显示出来一个跳动的时间，由于这些代码都放到了函数中，如果不调用，这些代码就像不存在一样，所以在函数定义的外面，我们要写调用语句。其中关键的代码是 setTimeout()，而需要再次调用的函数被写成了字符串，这说明 1000 毫秒后的调用，委托给了浏览器实现。

我们能不能将这个跳动的时间放到网页中呢？document.write 做不到，因为这个指令，如同临时编写的网页一样，动态的效果要求能够抹除原有内容，写上新的内容，但是 document.write 在默认情况下会遵守 HTML 中的文本流，新写的信息，不会覆盖原有的信息，而会出现在原有信息的下方，你可以将上述代码中要显示在状态栏中那句话替换成 document.write 来试试效果。你会发现调用一次后，便会出错。

这时我想到了表单中的输入框，输入框基本符合我的要求，问题是如何用 JavaScript 访问输入框呢？测试下面的代码。

```
<html>
    <body>
        <form name="f1">
            <input type="text" name="time"/>
        </form>
        <script>
            f1.time.value = "这里可以写入时间";
        </script>
    </body>
</html>
```

首先发现 JavaScript 的代码实际上可以嵌入到 HTML 中任何部分，在我们的这个应用中，我只能写在这个位置，因为我要使用那个输入框，如果写在输入框的上面，输入框对象还不存在，就没法进行这样的操作了。

然后发现给 form 命名为 f1，输入框命名为 time，这样就可以通过 f1.time，访问这个输入框，value 是这个输入框的属性，放到等号的左边，我们可以设置内容到这个输入框，放到等号的右边，就可以取得输入框中的内容了。

测试没问题了，你便可以让时间跳动在输入框中了。默认的输入框长度可能装不下全部的时间信息，别忘了可以设置 size 属性。

能够将时间设置到输入框中，已经接近完美了，我们完全可以通过设置取消的输入框的边框，这样看上去好像只是时间跳动在网页上。

其实还可以直接将时间显示在网页上，使用 div。你可能会想到 div 不是配合 CSS 使用的吗？没错，我们还需要 CSS，因为 JavaScript 能够操作 CSS 中所有的属性，所以我们要用 HTML+CSS+JavaScript 结构来显示时间。测试下面的代码。

```
<html>
    <body>
        <div id="time"/>
        <script>
            document.getElementById("time").innerHTML = "显示内容到 div 中";
        </script>
    </body>
</html>
```

注意大小写。还有个小问题，这样写代码不太舒服，如果网页文件中内容很多，JavaScript 代码就会这一块那一块的散落在 HTML 中，这样无论是程序员还是美工都会感到头痛，所以人们常常这样来写。具体代码如下所示。

```
<html>
    <head>
        <script>
        function writeDiv() {
            document.getElementById("time").innerHTML = "显示内容到 div 中";
        }
        </script>
    </head>
    <body onLoad="writeDiv()">
        <div id="time"/>
    </body>
</html>
```

on 开头的 HTML 属性是 JavaScript 的事件，如同 Java，也有鼠标、键盘、单击这样的事件，在这个任务中，我不深入，只是使用了<body>标记的 onLoad 事件，这样做的好处是，onLoad 事件开始的时候，body 中所有的对象就都已经准备好了。

因为 onLoad 是 HTML 的属性，所有 onLoad 不区分大小写，但是后面的值是 JavaScript 代码，需要区分大小写。

2.7.7 漂浮的时间显示

前面提到，JavaScript 完全能够操作 CSS 的属性，现在我们来体会这样的操作，基本做法是用 JavaScript 找到页面中的对象，我们一直用 style 来说明 CSS 语句，所以找到对象后就去获得

style 属性，然后获得 CSS 属性，就是 CSS 文档中提到的那些属性，但是很多 CSS 属性中间包含“-”，可不能在 JavaScript 的名称中使用“-”，所以遇到“-”，将这个符号去掉，减号是为了连接前后两个单词的，现在直接将两个单词写到一起，第二个单词的第一个字母大写就好了，例如，CSS 的属性 font-size，在 JavaScript 中就写成 fontSize。

测试下面的代码，感受一下操作过程。

```
<html>
    <head>
        <script>
            function boldMess() {
                document.getElementById("mess").style. fontWeight="bold";
            }
        </script>
    </head>
    <body onLoad="boldMess()">
        <div id="mess">一段文字</div>
    </body>
</html>
```

文字的加粗是靠 JavaScript 的语句完成的。这个案例比较简单，我们知道可以设置一个 div 中的内容是“绝对定位”，我们还可以用 setTimeout 产生动画效果，绝对定位的情况下，我们可以设置 top、left 这样类似于坐标的值，那么我们应该能够像 Java 课程中小球案例一样，让文字在网页上移动。你完全能够自己尝试着实现这样的效果，具体代码如下。

```
<html>
    <head>
        <script>
            x = 30;
            y = 30;
            function boldMess() {
                document.getElementById("mess").style.left=x+"px";
                document.getElementById("mess").style.top=y+"px";
                x ++;
                y ++;
                setTimeout("boldMess()",100);
            }
        </script>
    </head>
    <body onLoad="boldMess()">
        <div id="mess" style="position:absolute;">一段文字</div>
    </body>
</html>
```

怎么样，能让“一段文字”向右下角移动吧。我们还可以试试撞墙反弹的效果，这样我们就要知道墙在什么地方，也就是说要知道浏览器的大小，可以先到文档中找找看。

尝试着实现这个效果并不容易，一方面需要了解 JavaScript 对象的属性都有什么，这个可以在

JavaScript 文档中查找，一方面需要了解 CSS 属性设置，还要理清反弹的逻辑，真正头疼的是，如果你写错了不会报错，好在浏览器上定位的错误行号通常是准的，所以建议写一点测试一点，多利用 alert 工具。具体代码如下。

```
<html>
    <head>
        <script>
            function init() {
                f = 0;//方向
                x = 30;
                y = 30;
                //body 的宽度和高度
                width = document.body.clientWidth;
                height = document.body.clientHeight;
                //文字的长度
                divWidth = document.getElementById("mess").clien tWidth;
                divHeight = document.getElementById("mess").clien tHeight;
                myMove();
            }
            function myMove() {
                document.getElementById("mess").style.left=x+"px";
                document.getElementById("mess").style.top=y+"px";
                //设置飞行姿态
                if(f==0) {
                    x ++;
                    y ++;
                }
                if(f==1) {
                    x --;
                    y ++;
                }
                if(f==2) {
                    x --;
                    y --;
                }
                if(f==3) {
                    x ++;
                    y --;
                }
                //改变飞行姿态
                if((x+divWidth)>width) {
                    if(f==0) {
                        f = 1;
                    }else {
                        f = 2;
                    }
                }
                if((y+divHeight)>height) {
                    if(f==1) {
                        f = 2;
```

```
                }else {
                    f = 3;
                }
            }
            if(x<0) {
                if(f==2) {
                    f = 3;
                }else {
                    f = 0;
                }
            }
            if(y<0) {
                if(f==3) {
                    f = 0;
                }else {
                    f = 1;
                }
            }
            setTimeout("myMove()",10);
        }
    </script>
</head>
<body onLoad="init()">
    <div id="mess" style="position:absolute;">一段文字</div>
</body>
</html>
```

我将代码写在两个函数中，一个是 init，负责初始化获得一些基本信息，在 JavaScript 中，没有全局变量和局部变量的区分，如果不是 init 函数的定义，以及通过 onLoad 调用，div 信息是得不到的。

熟练掌握了这个代码后，你就可以用跳动的时间显示替换 div 中的“一段文字”了。

还有一件事，就是得到 div 对象的语句是 document.getElementById("mess")，这是个标准的写法，这个写法源于 XML，但是在 HTML 中有个简化的写法，比如，定义 top 值：mess.style.top="100px" ;，这样就可以了，但 getElementById 也是有用的，我希望你现在就练熟它。

不知道你自己有没有意识到，我们可以做很多动画了，甚至漫天星星，下大雪用 JavaScript 实现在网页中也是没有问题的。我提示一下，想必你不会在写 300 个套在 div 标记里的*号，你完全可以借用 JavaScript 的循环，加上 document.write 产生 300 个星星。

2.8　再看搜狐邮箱的用户登录

之前在实现这个用户登录的时候，我们忽略了其中的动作，要说搜狐邮箱用户登录的动作非常简单，就是单击登录前检查一下，如果密码是空的话，会提示密码不能为空，如果没有 JavaScript，这个问题会送到 Tomcat 那里，由后台的程序检查，并且将结果送回浏览器，这是很

慢的设计，现在 JavaScript 完全可以在前端检查。

为了简化代码，我不仔细实现显示效果，只是演示密码为空的检查动作程序，希望你能在此基础上实现出完美的用户登录页面。

我们测试一下真正的搜狐邮箱登录界面，发现单击登录按钮的时候，JavaScript 会开始检查密码栏目，登录按钮是 submit 按钮，自身有提交的功能，也就是说，要在提交动作前，进行检查，如果有问题就不提交了，那么我们就不能使用 submit 按钮，因为 submit 按钮是一定要提交的，我现在先用 button 按钮，button 只是显示按钮的样子，没有提交的功能。

下面我将简化后的用户登录页面列举出来。具体代码如下。

```
<html>
    <body>
        <form name="f1">
            邮 箱：<input type="text" name="user"/><br/>
            密 码：<input type="password" name="pass"/>
            <a href="">忘记密码</a>
            <br/>
            <input type="checkbox" name="save"/>记住登录状态
            <a href="">https 安全访问</a><br/>
            <input type="button" value="登录"/>
        </form>
    </body>
</html>
```

测试一下，显示效果自然是不堪入目，美化的工作由你来完成，我只是在此基础上实现功能。

验证密码的动作是登录按钮按下的时候开始进行的，和 Java 大同小异，我们将这叫做事件，比 Java 简单多了，JavaScript 的事件都是 on 开头的属性，你可以在帮助文档中查找属性来了解 JavaScript 都有什么事件，我们在此之前用过 onLoad 事件，现在对于按钮，我们用的是 onClick 事件，事件的处理程序通常是一个 JavaScript 函数，对于非常简单的事件处理，比如，单击的时候出现一个警告框，直接将 JavaScript 语句放到 onClick 属性后面也可以，但是毕竟这样能做的操作很少，所以也就很少这样使用，通常只是在编程测试的时候用一下。

```
<input type="button" value="登录" onClick="alert('测试')"/>
```

你注意一下，因为 JavaScript 语句放到了属性的值中，而属性的值又被双引号包围，所以 JavaScript 语句中就不能使用双引号了，我们用单引号代替，好在 HTML 和 JavaScript 都是非常宽松的技术，所以并没有十分严格的规定，字符串是用双引号还是用单引号定义。

现在测试一下，有效果后再向下继续。我只写出 JavaScript 的事件处理函数和事件定义的代码。具体代码如下。

```
<script>
    //按钮事件处理函数
    function valiPass() {
        alert("响应登录按钮");
```

```
    }
</script>
```

下一句话是定义按钮事件的处理函数。

```
<input type="button" value="登录" onClick="valiPass()"/>
```

下面我们要取得密码输入框中的值，然后进行判断。form 表单已经定义了名字是 f1，而密码输入框的名字是 pass，那么输入框的值就是 f1.pass.value，通过这个属性，你可以取用户输入的值，也可以设置这个值。验证函数的判断代码如下。

```
function valiPass() {
    if(f1.pass.value==""){
        alert("密码不能为空");
    }
}
```

上述代码测试通过后，我们再看搜狐的做法，它是将提示信息以红色显示在上面，现在我们已经能够使用 div 来输出信息了。

```
<html>
    <script>
    function valiPass() {
        if(f1.pass.value==""){
            mess.innerHTML="请输入通行证密码";
            f1.pass.focus();
        }else {
            mess.innerHTML="";
            f1.submit();
        }
    }
    </script>
    <body>
        <form name="f1">
            <div id="mess" style="color:red;"></div>
            邮 箱: <input type="text" name="user"/><br/>
            密 码: <input type="password" name="pass"/>
            <a href="">忘记密码</a>
            <br/>
            <input type="checkbox" name="save"/>记住登录状态
            <a href="">https 安全访问</a><br/>
            <input type="button" value="登录" onClick="valiPass()"/>
        </form>
    </body>
</html>
```

添加的内容是，在提示信息显示的位置，加入一个 div，没有内容，字体颜色是红色，更加细致的样式，你可以自己添加。

在事件处理函数中，判断 f1.pass.value 是否是空字符串，如果是，那么写提示信息到 div 中，并且用 f1.pass.focus()方法，将输入焦点放到密码输入框中。如果密码不是空的，那么清除 div 的内

容，这一步看似不需要，其实是优秀程序员的良好习惯，防止用户在意外的情况下看到遗留的提示信息。f1.submit()方法实现了提交的功能，button 按钮也就等同于 submit 按钮了。

我们将按钮改成 image，发现这段代码不能像我们想象的那样工作，因为 image 拥有 submit 的功能，无论你做了什么样的操作，最后都会提交，由于我们后台没有响应的程序，所以提交产生了刷新页面的效果，问题是如果密码为空我们并不想提交。解决办法是在不想提交的判断中 return false，而 onClick 事件中也使用 return。当然如果验证没问题，也就不需要 f1.submit()操作了。

```
<html>
    <script>
    function valiPass() {
        if(f1.pass.value=="") {
            mess.innerHTML="请输入通行证密码";
            f1.pass.focus();
            return false;
        }else {
            mess.innerHTML="";
            return true;
        }
    }
    </script>
    <body>
        <form name="f1" action="">
            <div id="mess" style="color:red;"></div>
            邮 箱：<input type="text" name="user"/><br/>
            密 码：<input type="password" name="pass"/>
            <a href="">忘记密码</a>
            <br/>
            <input type="checkbox" name="save"/>记住登录状态
            <a href="">https 安全访问</a><br/>
            <input type="image" src="button_login.gif" onClick= "return
valiPass()"/>
        </form>
    </body>
</html>
```

事实上我们还可以在 form 标签中使用 onSubmit 事件，你可以自己试着实现一下。

我们发现在搜狐邮箱的用户登录页面中，如果网速慢的话，单击登录按钮后，会看到一段提示的话，现在我来实现一下。具体代码如下。

```
<html>
    <script>
    function valiPass() {
        if(f1.pass.value=="") {
            mess.innerHTML="请输入通行证密码";
            f1.pass.focus();
            return false;
        }else {
```

```
            mess.innerHTML="";
            loginMsg.style.display="block";
            login.style.display="none";
            setTimeout("mySubmit()" , 3000);
            return false;
        }
    }
    function mySubmit() {
        f1.submit() ;
    }
    </script>
    <body>
        <form name="f1" action="">
            <div id="mess" style="color:red;"></div>
            邮 箱：<input type="text" name="user"/><br/>
            密 码：<input type="password" name="pass"/>
            <a href="">忘记密码</a>
            <br/>
            <input type="checkbox" name="save"/>记住登录状态
            <a href="">https 安全访问</a><br/>
            <div id="login">
<input type="image" src="button_login.gif" onClick="return valiPass()"/>
            </div>
            <div id="loginMsg" style="color:red;display:none;">
                正在登录搜狐通行证，请稍候...
            </div>
        </form>
    </body>
</html>
```

这样能够产生同样的效果，但是我们知道真正的网页代码不可能是这个样子的，一定是在单击登录按钮后，便显示提示信息，一直到服务器返回结果，而实现这个效果的过程很有意思，显示提示信息其实有两种办法，一是用 innerHTML 的属性赋值，甚至进而修改 CSS 属性以便达到邀请的效果。

现在我用的办法是，定义一个隐藏的 div，准备好提示信息，在需要的情况下，取消隐藏，这样 JavaScript 的编码相对容易，常常被用于需要显示复杂内容的情况。

2.9　京东商城的新用户注册

表单验证是作为 Java Web 程序员非常重要的任务，而用户登录的验证相对简单，所以我们现在来练习一下更复杂的用户注册页面，为了避免恶意注册，现在的注册页面都有一项是验证码，验证码的原理是在服务器中随机生成一个字符串，然后用这个字符串生成包含干扰的图片，确保图片无法被软件自动识别，用户看到这个图片后输入图片信息，服务器比对后确定这是人工

输入，如果没有验证码，我们就可以用程序进行自动注册或登录，这样的行为通常是恶意的。但是因为到目前为止，我们还没有学如何写服务器端的程序，所以没法实现验证码的功能，因此在这个任务中，我暂时取消验证码，等到你有了这个能力后，再加上这个功能。

这个任务对于现在的你来说，相当复杂，至少需要 300 行语句，如果 300 行语句是 Java 的也没什么，问题是我们面对的是宽松的 HTML、没有错误提示的 JavaScript 和复杂的 CSS，三种语法混合本身就不容易。

强烈建议你在 JavaScript 中学会有效地利用 alert，这个语句不但能够告诉你变量的值，告诉你某个对象是否存在，还可以用来验证程序流程是否是你设想的那样。当效果没有出来，到 IE 浏览器的左下角看看，有没有叹号，如果有的话，双击叹号，虽然通知给你的错误信息不是太详细，但是在这些技术的环境下，也算是难得的帮助了。当然真正起作用的是你的态度，排除错误完成任务的决心和不断探索的精神和专注力。

如图 2-30 所示为京东商城用户注册页面。

图 2-30

因为我们现在学习的重点在于 JavaScript 的动作，所以我不特别关心美观问题，提交按钮取消了图片的设置，另外用户注册协议的内容过长，而且与我们现在要练习的代码关联不大，所以在我实现的时候将忽略用户注册协议，HTML 部分只实现上面图片的功能（去掉验证码），京东商城的页面动作请具体参考它的网站。

下面的代码提供了初次访问的页面样式。

```
<html>
<style type="text/css">
.label{
```

```
    position:absolute;
    right:70%;
}

.fi{
    position:relative;
    left:30%;
}

.clr{
    height:20px;
}

.s{
    font-weight:lighter;
    color:red;
}

.text{
    font-family:宋体;
    width:200px;
}

.v{
    color:cccccc;
    font-size:12px;
}

a{
    font-size:12px;
}

.btn-img{
    position:relative;
    left:30%;
}
</style>
<body >
    <form id="f1">
              <div class="label"><b class="s">*</b>用户名：</div>
              <div class="fi">
                    <input  type="text"  name="username"  class="text"
tabindex="1"/>
                    <br/><div class="clr"></div>
          </div>

           <div class="label"><b class="s">*</b>设置密码：</div>
          <div class="fi">
```

```
                <input    type="password"    name="pwd"    class="text"
tabindex="2"/>
             
                <input type="checkbox" name="visi" id="viewpwd"/>
                 <label class="v">显示密码字符</label><br/>
                 <div class="clr"></div>
          </div>

         <div class="label"><b class="s">*</b>确认密码：</div>
         <div class="fi">
             <input type="password" name="pwd2" class="text" tabindex
="3"/>
             <br/><div class="clr"></div>
         </div>

       <div class="label"><b class="s">*</b>邮箱：</div>
       <div class="fi">
             <input      type="text"      name="mail"      class="text"
tabindex="4"/>
                
               <label class="v">免费邮箱：</label>
          <a href="">搜狐</a>
          <a href="">网易</a><br/>
             <div class="clr"></div>
       </div>

       <div class="label">推荐人用户名：</div>
       <div class="fi">
            <input type="text" name="referrer" class="text"
value="可不填" tabindex="5"/>
           <br/>
            <div class="clr"></div>
          </div>
       <input type="button" class="btn-img" id="registsubmit"
value="同意以下协议，提交" tabindex="8"/>
   </form>
   </body>
</html>
```

这些代码没有功能，只是样子，显示的效果如图 2-31 所示。你不需严格按照我的代码来实现，只要大体能够产生这样的效果就可以了。

现在我们在上面添上功能，为了便于理解，我一个一个输入框来实现，提供的代码也将是片段。

当光标到用户名后面的输入框时，我们看到下面会出现一句提示的话，如果输入用户名不符合要求，就会出现错误提示，如果正确输入框后面会出现一个绿色的对勾。如图 2-32 所示。

图 2-31

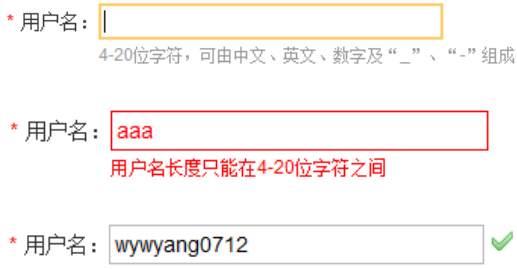


图 2-32

以上的三幅图是这个输入框的三个状态，我们用 div 显示提示信息的时候，有两种常用的方式，一是事先写好预设的信息到 div，将其设置为隐藏，当事件发生时，将相应的信息设置成可见。第二种方式是直接用 innerHTML 写信息到 div 中，为了更好地学习，我在实现的时候，混合使用两种方式。

先预设，4-20 位字符，可由中文、英文、数字及“_”、“-”组成，并且将这句话设置成隐藏，下面是表单部分添加的 HTML 代码。

```
<div class="fi">
    <input type="text" name="username" class="text" tabindex="1"/><br/>
    <div id="username_mess" class="clr">4-20 位字符，可由中文、英文、数字及
“_”、“-”组成</div>
</div>
```

在这个 div 上我同时使用了 id 和 class，JavaScript 选取 id 是非常容易的，考虑到下面几个提示信息的操作是不同的，所以接下来我们要设置不同的 id。但是考虑到提示信息的颜色、字体大小等样式是相同的，所以我将为所有显示提示信息的 div 设定一个 class 名，这样就能通过一次设定，统一所有的提示样式了。

下面是 CSS 部分的代码。

```
.clr{
    height:20px;
    color:AAAAAA;
```

```
    font-size:12px;
    visibility:hidden;
}
```

现在来看动作，光标进入输入框的事件是 onFocus，而离开输入框的事件是 onBlur，我们先来实现第一个效果，HTML 部分的代码如下。

```
<div class="fi">
<input type="text" name="username" class="text" tabindex="1"
  onFocus="inUser()" onBlur="outUser()"/><br/>
<div id="username_mess" class="clr">4-20 位字符，可由中文、英文、数字及
“_”、“-”组成
    </div>
</div>
```

JavaScript 部分的代码如下。

```
<script>
    function inUser() {
        username_mess.style.visibility="visible";
    }
    function outUser() {
        username_mess.style.visibility="hidden";
    }
</script>
```

这样第一幅图的提示信息功能就完成了，我们再来分析第二幅和第三幅图的功能，我们发现当用户名输入框失去焦点的时候，会有三种情况，如果输入框中什么都没有是空的，那么所有的提示信息都会消失，我们其实已经实现了这个效果。如果用户输入的名字位数小于 4 位，或是大于 20 位，那么会出现第二幅图的错误信息。如果用户输入的正确，那么会在输入框的后面出现绿色的对钩。

这样看来，第三个正确信息和前两个的位置是不同的，而且相对固定，我们完全可以预设对钩的图片，默认为隐藏，需要的情况下显示就是了。

无论是那种情况，看来我们都要做判断，判断的条件是字符串的长度。在 JavaScript 中字符串是个对象，它有些成员方法，你现在可以查看一下帮助文档，寻找字符串项目，然后根据我的这些提示，实现这些功能。

以下是我所实现的代码，再次说明，你的代码不需要和我的一模一样，只要能够实现这个功能就好了。

我们可以在输入框代码的后面跟上这句话。

```
<label id="username_ok" class="ok"><img src="ok.jpg"></label>
```

没有使用 div，而是用了 label，因为 label 不会在新的一行显示，当然即便用 div，也完全可以通过 CSS 代码让这个图片显示在输入框后面，但是毕竟比使用 label 要麻烦一点。

你发现我在这个标记中，即设定了 id，也提供了 class，这是因为我发现在这个项目中，有好多处都有显示对钩的需要，这样我就有统一设置这些对钩的可能，但是操作对钩，比如，显示

还是不显示，是需要 JavaScript 来选取的，所以 id 是给 JavaScript 准备的。

既然需要很多对钩，又会设置统一的 class，我就没有必要写很多的 img 了，统一用 CSS 设置图片显示就行了，所以这段 HTML 代码进一步被简化成了如下代码。

```
<label id="username_ok" class="ok"></label>
```

对应的 CSS 代码如下，你可以先自己查文档试着实现。

```
.ok{
    background:url("ok.jpg");
    width:17px;
    height:16px;
    visibility:hidden;
}
```

当然，要想看到这个对钩，需要将最后一句隐藏设置先去掉，在你确保对勾图片存在的情况下再加上这句话。上面的 CSS 代码实现起来并不容易，经过一番探索，你会发现，如果不设置宽和高，就没法显示。

2.9.1　String 对象操作

现在我来实现功能，以下是 JavaScript 部分代码。

```
function outUser() {
    if(f1.username.value==""){
        username_mess.style.visibility="hidden";
    }else {
        u = f1.username.value;
        if(u.length<4||u.length>20) {
            username_mess.innerHTML="用户名长度只能在 4-20 位字符之间";
            username_mess.style.color="red";
            f1.username.style.color="red";
            f1.username.style.border="1px solid red";
            username_ok.style.visibility="hidden";
        }else {
            username_ok.style.visibility="visible";
            username_mess.innerHTML="";
            f1.username.style.color="black";
            f1.username.style.border="1px solid AAAAAA";
        }
    }
}
```

不知道我实现的代码是不是比你实现的复杂，我们不但要实现自己想要的功能，很多细节的操作也是不可缺少的，比如，输入错误会造成很多显示的颜色为红色，一旦正确了，这些颜色要还原回去。

其实还有一种错误，如图 2-33 所示。

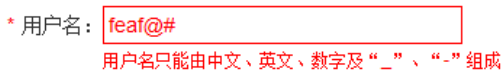


图 2-33

要判断用户名只能是中文、英文、数字、下划线和减号。

```
b = true;
for(i= 0; i < u.length; i ++) {
        c = u.charAt(i);
        if(!((c>='a' && c <='z')||(c>='A'&&c<='Z')||(c>='0'&&c<='9')|
|c=='_'||c =='-')) {
       b = false;
       break;
    }
}
```

上面的代码判断了英文、数字、下划线和减号，下面的程序将根据 b 的值来确定是否符合规则，判断中文是难点。

通过上网查询，我得知最小的汉字编码是 19968，最大的是 40623，但是 charAt 得到的不是编码，而是 string（在 JavaScript 中没有 Char），查询帮助文档得知还有一个方法是 charCodeAt 得到的是编码，修改程序后代码如下。

```
b = true;
for(i= 0; i < u.length; i ++) {
    c = u.charAt(i);
    cc = u.charCodeAt(i);
    if(!((c>='a' && c <='z')||(c>='A'&&c<='Z')||(c>='0'&&c<='9')||c==
'_'||c=='-'||
                (cc>=19968&&cc<=40623))) {
       b = false;
       break;
       }
    }
```

2.9.2 正则表达式

上面那种一位位的判断确实很麻烦，以后可能还要遇到比这还复杂的规则，事实上大多数的项目中都会有大量输入验证的工作，为此出现了一个叫做“正则表达式”的技术，目前大多数计算机语言都支持这项技术。

正则表达式是通过一个表达式来设定正确的规则，验证的时候只需要询问输入的字符串是否符合这个表达式就可以了。

因为正则表达式不是 Java 或 JavaScript 发明的，所以对于我们这样的程序员，这个表达式的语法看起来很古怪，以至于有人一直回避正则表达式，但是正则表达式的使用确实能够大幅度地

降低代码难度。

我们先来看只允许中文字符的正则表达式写法：^[\u4e00-\u9fa5]{4,20}$。

是不是看上去很古怪，我们一点点来分析，^和$在正则表达式中分别代表开头和结尾，如果在表达式的前后同时使用，意味着分析字符串的开头到结尾只能是表达式中描述的内容，这两个符号事实上是现在这个用户名判断的关键。

[]的目的是设定匹配的什么，如果写[abc]那么字符串中只允许有 abc 这三个字母，如果写成[a-z]就允许所有的小写字母，在上面的表达式里面[]中的内容说明的是中文。

{4,20}表示允许的位数，这里允许 4 到 20 位字符，我将允许的字符和位数写到一起是为了展示正则表达式的能力，而在我们的这个项目中位数错误和内容错误显示的错误提示是不同的，所以我们会分开来写两个正则表达式。

我先演示如何在 JavaScript 中使用正则表达式，然后再继续研究。这些代码写在 HTML 中如下所示。

```
<script>
    str = "搜狐邮箱用户登录";
    reg = /^[\u4e00-\u9fa5]{4,20}$/;
    alert(reg.test(str));
</script>
```

你看到 reg 的赋值操作，要注意的是正则表达式不是字符串，不能使用双引号，而是用两个/定义表达式，这样 reg 就有 test 方法了，返回值是 true 和 false，另外还有 exec 方法，返回值是 null，意味着不匹配，以及字符串，意味着匹配上了什么字符串。

上面的代码可以用来帮助练习我讲的内容，你可以改变 str 的内容，测试匹配还是不匹配，比如，上面的代码就会得到 true 的显示，如果你修改 str="建构式 Java Web 教程"，那么返回值就是 false，因为其中包含了英文字母。

上述表达式只允许中文字符，我们在这个基础上增加对其他字符的允许。

```
^[\u4e00-\u9fa5a-z A-Z]{4,20}$
```

这样“建构式 Java Web 教程”就会测试为 true。要知道在 a-z 和 A-Z 之间我放了一个空格，这不是为了看着舒服，而是用来说明，我允许有空格，如果不允许有空格，那么就将所有的内容连在一起写。

\d 是允许数字，\D 是不允许数字，这个定义看起来是真气人，-和_只要写到中括号里面就行了。最终的表达式是：^[\u4e00-\u9fa5a-z A-Z\d-_]{4,20}$。

而\w 能够替代[a-zA-Z0-9_]，所以用\w 替换后，表达式是：^[\u4e00-\u9fa5 \w-]{4,20}$

这样在项目中 JavaScript 的代码就改变成如下形式。

```
function outUser() {
    u = f1.username.value;
    f1.username.style.border="1px solid AAAAAA";
    if(u==""){
```

```
            username_mess.style.visibility="hidden";
            return;
        }

        reg = /.{4,20}/;
        if(!reg.test(u)) {
            username_mess.innerHTML="用户名长度只能在 4-20 位字符之间";
            username_mess.style.color="red";
            f1.username.style.color="red";
            f1.username.style.border="1px solid red";
            username_ok.style.visibility="hidden";
            return;
        }

        reg = /^[\u4e00-\u9fa5 \w-]{4,20}$/;
        if(reg.test(u)) {
            username_ok.style.visibility="visible";
            username_mess.innerHTML="";
            f1.username.style.color="black";
        }else {
            username_mess.innerHTML="用户名只能由中文、英文、数字、_和-组成";
            username_mess.style.color="red";
            f1.username.style.color="red";
            f1.username.style.border="1px solid red";
            username_ok.style.visibility="hidden";
        }
    }
```

有了上述案例的基本体验，下面我列举一下正则表达式中会出现的规则，如表 2-1 所示，这个表格不需要你记住，需要的时候来查就好了，不过现在至少将每个项目都仔细地看一遍，头脑中有个印象。

表 2-1

字符	描述
\	转义符。例如，'n'匹配字符"n"。'\n'匹配一个换行符。序列'\\'匹配"\"而"\("则匹配"("。
^	匹配输入字符串的开始位置。如果设置了 RegExp 对象的 Multiline 属性，^也匹配'\n'或'\r'之后的位置。
$	匹配输入字符串的结束位置。如果设置了 RegExp 对象的 Multiline 属性，$也匹配'\n'或'\r'之前的位置。
*	匹配前面的子表达式零次或多次。例如，zo*能匹配"z"以及"zoo"。*等价于{0,}。
+	匹配前面的子表达式一次或多次。例如，'zo+'能匹配"zo"以及"zoo"，但不能匹配"z"。+等价于{1,}。
?	匹配前面的子表达式零次或一次。例如，"do(es)?"可以匹配"do"或"does"中的"do"。?等价于{0,1}。
{*n*}	*n* 是一个非负整数。匹配确定的 *n* 次。例如，'o{2}'不能匹配"Bob"中的'o'，但是能匹配"food"中的两个 o。
{*n*,}	*n* 是一个非负整数。至少匹配 *n* 次。例如，'o{2,}'不能匹配"Bob"中的'o'，但能匹配"foooood"中的所有 o。'o{1,}'等价于'o+'。'o{0,}'则等价于'o*'。
{*n*,*m*}	*m* 和 *n* 均为非负整数，其中 *n*<=*m*。最少匹配 *n* 次且最多匹配 *m* 次。"o{1,3}"将匹配"fooooood"中的前三个 o。'o{0,1}'等价于'o?'。请注意在逗号和两个数之间不能有空格。

续表

字符	描述
?	当该字符紧跟在任何一个其他限制符(*,+,?,{n},{n,},{n,m})后面时，匹配模式是非贪婪的。非贪婪模式尽可能少地匹配所搜索的字符串，而默认的贪婪模式则尽可能多地匹配所搜索的字符串。例如，对于字符串"oooo"，'o+?'将匹配单个"o"，而'o+'将匹配所有'o'。
.	匹配除"\n"之外的任何单个字符。要匹配包括'\n'在内的任何字符，请使用像'[.\n]'的模式。
(*pattern*)	匹配 *pattern* 并获取这一匹配。所获取的匹配可以从产生的 Matches 集合得到。要匹配圆括号字符，请使用'\('或'\)'。
(?:*pattern*)	匹配 *pattern* 但不获取匹配结果，也就是说这是一个非获取匹配，不进行存储供以后使用。这在使用"或"字符(|)来组合一个模式的各个部分是很有用的。例如，'industr(?:y|ies)就是一个比'industry|industries'更简略的表达式。
(?=*pattern*)	正向预查，在任何匹配 *pattern* 的字符串开始处匹配查找字符串。这是一个非获取匹配，也就是说，该匹配不需要获取供以后使用。例如，'Windows(?=95|98|NT|2000)'能匹配"Windows 2000"中的"Windows"，但不能匹配"Windows3.1"中的"Windows"。预查不消耗字符，也就是说，在一个匹配发生后，在最后一次匹配之后立即开始下一次匹配的搜索，而不是从包含预查的字符之后开始。
(?!*pattern*)	负向预查，在任何不匹配 Negative lookahead matches the search string at any point where a string not matching *pattern* 的字符串开始处匹配查找字符串。这是一个非获取匹配，也就是说，该匹配不需要获取供以后使用。例如'Windows(?!95|98|NT|2000)'能匹配"Windows3.1"中的"Windows"，但不能匹配"Windows2000"中的"Windows"。预查不消耗字符，也就是说，在一个匹配发生后，在最后一次匹配之后立即开始下一次匹配的搜索，而不是从包含预查的字符之后开始
x|*y*	匹配 *x* 或 *y*。例如，'z|food'能匹配"z"或"food"。'(z|f)ood'则匹配"zood"或"food"。
[*xyz*]	字符集合。匹配所包含的任意一个字符。例如，'[abc]'可以匹配"plain"中的'a'。
[^*xyz*]	负值字符集合。匹配未包含的任意字符。例如，'[^abc]'可以匹配"plain"中的'p'。
[*a-z*]	字符范围。匹配指定范围内的任意字符。例如，'[a-z]'可以匹配'a'到'z'范围内的任意小写字母字符。
[^*a-z*]	负值字符范围。匹配任何不在指定范围内的任意字符。例如，'[^a-z]'可以匹配任何不在'a'到'z'范围内的任意字符。
\b	匹配一个单词边界，也就是指单词和空格间的位置。例如，'er\b'可以匹配"never"中的'er'，但不能匹配"verb"中的'er'。
\B	匹配非单词边界。'er\B'能匹配"verb"中的'er'，但不能匹配"never"中的'er'。
\c*x*	匹配由 x 指明的控制字符。例如，\cM 匹配一个 Control-M 或回车符。*x* 的值必须为 A-Z 或 a-z 之一。否则，将 c 视为一个原义的'c'字符。
\d	匹配一个数字字符。等价于[0-9]。
\D	匹配一个非数字字符。等价于[^0-9]。
\f	匹配一个换页符。等价于\x0c 和\cL。
\n	匹配一个换行符。等价于\x0a 和\cJ。
\r	匹配一个回车符。等价于\x0d 和\cM。
\s	匹配任何空白字符，包括空格、制表符、换页符等。等价于[\f\n\r\t\v]。
\S	匹配任何非空白字符。等价于[^\f\n\r\t\v]。
\t	匹配一个制表符。等价于\x09 和\cI。
\v	匹配一个垂直制表符。等价于\x0b 和\cK。

续表

字符	描述
\w	匹配包括下划线的任何单词字符。等价于'[A-Za-z0-9_]'。
\W	匹配任何非单词字符。等价于'[^A-Za-z0-9_]'。
\x*n*	匹配 *n*，其中 *n* 为十六进制转义值。十六进制转义值必须为确定的两个数字长。例如，'\x41'匹配"A"。'\x041'则等价于'\x04'&"1"。正则表达式中可以使用 ASCII 编码。.
num	匹配 *num*，其中 *num* 是一个正整数。对所获取的匹配的引用。例如，'(.)\1'匹配两个连续的相同字符。
n	标识一个八进制转义值或一个后向引用。如果*n* 之前至少 *n* 个获取的子表达式，则 *n* 为后向引用。否则，如果 *n* 为八进制数字(0-7)，则 *n* 为一个八进制转义值。
nm	标识一个八进制转义值或一个后向引用。如果*nm* 之前至少有 is preceded by at least *nm* 个获取的子表达式，则 *nm* 为后向引用。如果*nm* 之前至少有 *n* 个获取，则 *n* 为一个后跟文字 *m* 的后向引用。如果前面的条件都不满足，若 *n* 和 *m* 均为八进制数字(0-7)，则*nm* 将匹配八进制转义值 *nm*。
nml	如果 *n* 为八进制数字(0-3)，且 *m* 和 *l* 均为八进制数字(0-7)，则匹配八进制转义值 *nml*。
un	匹配 *n*，其中 *n* 是一个用 4 个十六进制数字表示的 Unicode 字符。例如，\u00A9 匹配版权符号(?)。

为了更好地掌握正则表达式，我们再来理解一下 Email 地址验证的正则表达式，\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*

先来看@符号前面，\w 已经能够支持大多数用户的 Email 地址了，但是有些人会在注册邮箱的时候额外使用-、+或.，所以才有后面小括号的内容，甚至有些人的邮箱会有几个点，这样在小括弧后面使用*就没问题了。

邮箱地址在@后面通常一定要有点，所以写法更加复杂了，([-.]\w+)*运行在字母、数字和下划线的基础上有减号和点，但是这是允许，而不是强制，所以后面会跟\.这样就会确保在@后面一定会有一个点，这个点后面和前面的定义一样。

那么你自己尝试写出身份证号码的正则表达式，不允许 HTML 标记的正则表达式。你可以想到很多正则表达式的应用，不断地实战是学会正则表达式的唯一办法，网上也有很多资料，确保能够驾驭正则表达式后，在继续向下学习。

2.9.3 密码框验证

继续向下看密码的输入框，我们或许要将密码和确认密码两个部分和在一起实现，我还是先提供不同情况下的效果。

如图 2-34 所示，上面两幅图的效果，实现起来和用户名相应的效果雷同。

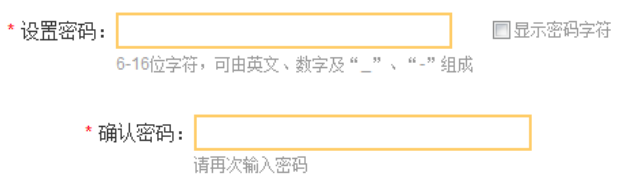


图 2-34

勾选显示密码字符复选框后，密码区域的底色变成粉红色，密码栏中的输入将显示内容。如图 2-35 所示。

图 2-35

这个效果不需过多解释，如图 2-36 所示为红色显示。

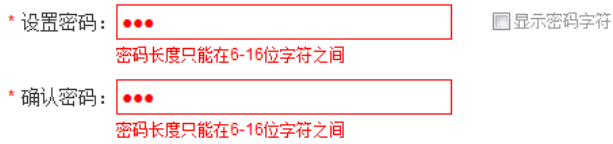


图 2-36

如图 2-37 所示这个效果比较特别，当输入字符达到 6 位，就会出现上面的显示，如果输入的内容只是数字或只是字母，或者只是下划线和减号，那么安全程度，后面弱的区域的底色会显示橙色，其余为灰色，混合两种类型的字符，安全程度的弱和中底色为橙色，三种类型的字符都有的情况下，三个区域都是橙色。

这个效果的确认在密码栏目上没有。输入正确，绿色对钩便会出现。

图 2-37

如图 2-38 所示为两次密码不一致的提示，如果正确，出现绿对钩。

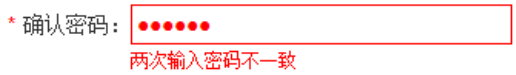


图 2-38

依照用户名的代码思路，我们先在 HTML 中准备对钩和提示信息。勾选显示密码字符复选框后，密码区域的底色变成粉红色，这个效果需要对两个密码输入框进行统一的操作，所以我在两个输入框区域外，加入了一个 div。具体代码如下。

```
<div id="pass">
    <div class="label"><b class="s">*</b>设置密码：</div>
    <div class="fi">
        <input type="password" name="pwd" class="text" tabindex="2"/>
           
```

```
        <input type="checkbox" name="visi" id="viewpwd"/>
        <label id="pass_ok" class="ok">6-16 位字符，可由英文、数字及"_"、"-
"组成</label>
        <label class="v">显示密码字符</label><br/>
            <div id="pass_mess" class="clr"></div>
        </div>

    <div class="label"><b class="s">*</b>确认密码：</div>
    <div class="fi">
        <input type="password" name="pwd2" class="text" tabindex="3"/>
        <label id="rpass_ok" class="ok"></label><br/>
        <div id="rpass_mess" class="clr">请再次输入密码</div>
    </div>
</div>
```

现在来实现第一组图片显示的功能。和用户名输入框的代码一样，需要分别定义获得焦点和失去焦点的事件。以下是 JavaScript 代码。

```
function inPass() {
    pass_mess.style.visibility="visible";
    f1.pwd.style.border="1px solid gold";
}

function outPass() {
    p = f1.pwd.value;
    f1.pwd.style.border="1px solid AAAAAA";
    if(p==""){
        pass_mess.style.visibility="hidden";
        return;
    }
}

function inRPass() {
    rpass_mess.style.visibility="visible";
    f1.pwd2.style.border="1px solid gold";
}

function outRPass() {
    p = f1.pwd2.value;
    f1.pwd2.style.border="1px solid AAAAAA";
    if(p==""){
        rpass_mess.style.visibility="hidden";
        return;
    }
}
```

对应的 HTML 代码也要修改，以便能够响应相关的事件。HTML 代码如下。

```
<input type="password" name="pwd" class="text" tabindex="2"
    onFocus="inPass()" onBlur="outPass()"/>
```

```
<input type="password" name="pwd2" class="text" tabindex="3"
    onFocus="inRPass()" onBlur="outRPass()" />
```

如果你已经能够实现第一组的功能，我们继续来看改变背景颜色的功能。

JavaScript 代码如下。

```
function visibe() {
       alert(f1.visi.checked);
}
```

配合 HTML 事件的调用。

```
<input type="checkbox" name="visi" id="viewpwd" onClick="visibe()"/>
```

测试上述代码，通过 checked 属性，能够得到这个复选框是否被选中的 boolean 值。在这个基础上，我们实现功能。设置背景颜色相对简单，但是如何能够实现将控制密码是否显示出来呢？理论上我们可以设置 type 属性。

```
f1.pwd.type="text";
```

但是经过测试发现这个设置在 IE 中是没有效果的，在火狐浏览器中可以，这是因为 type 属性在 IE 中是只读的，我们只能使用笨办法，将这个输入框放到一个 div 中，使用 JavaScript 重写这个 input。还有一个小问题，在网页中 div 会产生自动换行的效果，我们要使用 CSS 来处理这个问题。

JavaScript 代码如下。

```
function visibe() {
    if(f1.visi.checked){
        pass.style.backgroundColor="FFEEEE";
        v = f1.pwd.value;
        v2 = f1.pwd2.value;
        chang.innerHTML = "<input type='text' name='pwd' class='text'
tabindex='2' value='"+v+"' onFocus='inPass()' onBlur='outPass()'/>";
        chang2.innerHTML = "<input type='text' name='pwd2' class='text'
tabindex='2' value='"+v2+"' onFocus='inRPass()' onBlur='outRPass()'/>";
    }else {
        pass.style.backgroundColor="FFFFFF";
        v = f1.pwd.value;
        v2 = f1.pwd2.value;
        chang.innerHTML = "<input type='password' name='pwd' class='text'
tabindex='2' value='"+v+"' onFocus='inPass()' onBlur='outPass()'/>";
        chang2.innerHTML = "<input type='password' name='pwd2'
class='text' tabindex='2' value='"+v2+"' onFocus='inRPass()'
onBlur='outRPass()'/>";
    }
}
```

HTML 部分代码修改如下。

```
<div id="pass">
    <div class="label"><b class="s">*</b>设置密码：</div>
    <div class="fi">
```

```
        <div id="chang" style= "display:inline">
        <input  type="password"  name="pwd"  class="text"  tabindex="2"
onFocus="inPass()" onBlur="outPass()"/>
        </div>
           
        <input   type="checkbox"   name="visi"   id="viewpwd"   onClick=
"visibe ()"/>
        <label id="pass_ok" class="ok"></label>
        <label class="v">显示密码字符</label><br/>
        <div id="pass_mess" class="clr">6-16 位字符，可由英文、数字及"_"、
"-"组成
    </div>
    </div>

    <div class="label"><b class="s">*</b>确认密码：</div>
    <div class="fi">
        <div id="chang2">
        <input  type="password"  name="pwd2"  class="text"  tabindex="3"
onFocus="inRPass()" onBlur="outRPass()" />
        </div>
        <label id="rpass_ok" class="ok"></label><br/>
        <div id="rpass_mess" class="clr">请再次输入密码</div>
    </div>
</div>
```

在上面的代码中，不仅仅要处理显示的效果，还使用了代码逻辑，将用户已经输入的内容带入到新定义的输入框中。

下面我们来实现显示密码强度的功能。请先确保之前的代码你已经掌握了。

我们先实现让安全强度显示出来的功能，思路是在 HTML 中准备好这些内容，然后设置成隐藏状态，当输入 6 个字符的时候，用 JavaScript 将其显示出来。

HTML 代码如下。

```
<div id="strength">安全强度：
    <table class="showStren" align="center">
        <tr width="100px" align="center">
            <td id="l" class="ceil">弱</td>
            <td id="m" class="ceil">中</td>
            <td id="h" class="ceil">强</td>
        </tr>
    </table>
</div>
```

CSS 代码如下。

```
#strength{
    color:AAAAAA;
    font-size:12px;
    display:none;
```

```
}
.showStren{
    width: 135px;
    display: inline;
    color:white;
    font-size:12px;
}
.ceil{
    border:0.5px solid #FFFFFF;
    background-color:DDDDDD;
}
```

我们知道这个区域现在隐藏了两个 div，之前我们用的隐藏方式是 visibility:hidden;，这个隐藏方式虽然将内容隐藏掉了，但是位置还在，这样两个隐藏的 div 会让设置密码下方的空白过大，看上去不舒服，所以我们要使用 display:none;将其中一个 div 隐藏，以保证其中一个 div 暂时不占位置，具体的操作方法请查阅文档。

我们使用 onKeyup 事件来调用 JavaScript 处理程序。

```
<input type="password" name="pwd" class="text" tabindex="2"
onFocus="inPass()" onBlur="outPass()" onKeyup="keyPass()"/>
```

下面的 JavaScript 函数只是将安全强度显示处理。

```
function keyPass() {
    if(f1.pwd.value.length>=6){
        pass_mess.style.display="none";
        strength.style.display="inline";
    }
}
```

如果你已经能够实现上述功能的话，我们来看如何改变相应强度的颜色。我们需要先计算密码的强度，规则是将输入的字符分成三种类型，字母、数字、下划线和减号，如果密码中只有一种类型字符，密码强度就是弱，两种是中，三种都有就是强，我来写一个函数以便计算密码强度。具体代码如下。

```
function checkStrong(Str) {
    sum = 0;
    reg = /[a-zA-Z]/;
    if(reg.test(Str)) {
        sum ++;
    }
    reg = /\d/;
    if(reg.test(Str)) {
        sum ++;
    }
    reg = /[_-]/;
    if(reg.test(Str)) {
        sum ++;
    }
```

```
    return sum;
}
```

我们在事件处理程序 keyPass 中调用这个函数，并且根据返回值设定相应的背景颜色。具体代码如下。

```
function keyPass() {
    if(f1.pwd.value.length>=6){
        pass_mess.style.display="none";
        strength.style.display="inline";
        sum = checkStrong(f1.pwd.value);
        if(sum==1) {
            l.style.backgroundColor="orange";
        }
        if(sum==2) {
            l.style.backgroundColor="orange";
            m.style.backgroundColor="orange";
        }
        if(sum==3) {
            l.style.backgroundColor="orange";
            m.style.backgroundColor="orange";
            h.style.backgroundColor="orange";
        }
    }
}
```

最后判断两次密码是否一致，在 outRPass 函数中加入一个判断。具体代码如下。

```
if(!(f1.pwd.value==f1.pwd2.value)){
    rpass_mess.innerHTML="两次输入密码不一致";
    rpass_mess.style.color="red";
    f1.pwd2.style.color="red";
    f1.pwd2.style.border="1px solid red";
    rpass_ok.style.visibility="hidden";
    return;
}
```

2.9.4 邮箱地址验证

有了前面的代码和正则表达式的基础，邮箱验证相当的简单，你尝试后可以自己再看我的代码。

其中 HTML 部分代码如下。

```
<div class="label"><b class="s">*</b>邮箱：</div>
    <div class="fi">
          <input type="text" name="mail" class="text" tabindex="4"
                 onFocus="inMail()" onBlur="outMail()"/>
        <label id="mail_ok" class="ok"></label>
           
        <label class="v">免费邮箱：</label>
```

```
        <a href="">搜狐</a>
        <a href="">网易</a><br/>
        <div id="mail_mess" class="clr">请输入常用的邮箱，将用来找回密码、接
收订单通知等</div>
    </div>
</div>
```

有了前面的代码基础，CSS 几乎不用调整，JavaScript 部分代码如下。

```
function inMail() {
    mail_mess.style.visibility="visible";
    f1.mail.style.border="1px solid gold";
}
function outMail() {
    mm = f1.mail.value;
    f1.mail.style.border="1px solid AAAAAA";
    if(mm==""){
        mail_mess.style.visibility="hidden";
        return;
    }
    reg=/\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*/;
    if(reg.test(mm)) {
        mail_ok.style.visibility="visible";
        mail_mess.innerHTML="";
        f1.mail.style.color="black";
    }else {
        mail_mess.innerHTML="邮箱格式不正确";
        mail_mess.style.color="red";
        f1.mail.style.color="red";
        f1.mail.style.border="1px solid red";
        mail_ok.style.visibility="hidden";
    }
}
```

推荐人用户名的功能可自己来实现，输入框中的内容“可不填”是灰色的，如果光标进入这个输入框，那么“可不填”这句话会消失。

2.10　搜狐首页的菜单条

访问搜狐的首页，最上面会有一个会员登录的区域，如果登录成功那个区域会出现菜单条，鼠标放到一个项目上就会出现下拉菜单，我们的主要目的就是实现这个下拉菜单的功能。如图 2-39 所示，这个效果是鼠标放在消息上面的时候出现的。

我们先来实现出来没有弹出菜单的效果，里面有些图片，我想你自己一定能分析出来。下面是实现的代码。

图 2-39

```
<html>
    <style type="text/css">
    #title{
        background-color:azure;
        position:absolute;
        top:0;
        left:0;
        width: 103%;
        height:25px;
        border: 1px solid lightblue;
    }
    #item{
        position:absolute;
        left:300px;
    }
    ul{
        width:700px;
        font-size:12px;
    }
    li{
        list-style-type:none;
        float:left;
        margin:4px 10px;
    }
    #rr{
        background-color:red;
        width:12px;
        text-align:center;
        color:white;
    }
    </style>
    <body>
        <div id="title">
            <div id="item">
                <ul>
                    <li><b><font color="red">我的</font>搜狐</b></li>
                    <li><img src="nobody_v_110708.gif" width ="15px"/>
                        wymary@sohu.com
                        <img src="down.gif"/></li>
                    <li>个人中心</li>
                    <li>消息<font id="rr">8</font></li>
```

```
                <li>邮件<font id="rr">99+</font></li>
                <li>博客</li>
                <li>相册</li>
                <li>说两句</li>
                <li>应用<img src="down.gif"/></li>
            </ul>
        </div>
    </div>
</body>
</html>
```

鼠标放到上面的事件是 onMouseOver，我们的思路是提前准备好弹出的内容，并且隐藏起来，当鼠标到这个上面的时候，用 JavaScript 代码，将隐藏的内容显示出来。当 onMouseOut 事件产生，要重新将消息框隐藏起来，并且将消息恢复到原来的样子。这个任务的难点是，在鼠标离开消息区域后，又可能进入刚刚显示的消息框区域，这种情况下，消息框不能消失。

考虑到弹出消息框后，消息两个字的背景会变白色，8 那个数字的背景变成灰色，为了和显示出来的消息框融合的更恰当，HTML 的代码还要修改一下，请参考帮助文档来看我增加的代码，一句句的理解，一句句的在你的计算机上尝试，直到能够全部理解后，再删除掉全部代码，重新自己实现，直到了然于胸。具体代码如下。

```
<html>
<style type="text/css">
#title{
    background-color:azure;
    position:absolute;
    top:0;
    left:0;
    width: 103%;
    height:25px;
    border: 1px solid lightblue;
}
#item{
    position:absolute;
    left:300px;
}
ul{
    width:700px;
    font-size:12px;
}
li{
    list-style-type:none;
    float:left;
    margin:4px 10px;
}
.rr{
    background-color:red;
    width:12px;
```

```
        text-align:center;
        color:white;
    }
    #mess{
        position:absolute;
        z-index:10000;
        top:23px;
        left:632px;
        height:60px;
        width:200px;
        border:1px solid lightblue;
        display:none;
        font-size:12px;
    }
    #messItem{
        background-color:azure;
        position:relative;
        text-align:center;
        margin : 0px 4px;
        vertical-align: middle;
        padding-top:5px;
        width: 60px;
        height:23px;

    }
    </style>
    <script>
    function showMess() {
        messItem.style.backgroundColor="white";
        messItem.style.border="1px solid lightblue";
        mess.style.display="block";
        messItem.style.height="25px";
        messItem.style.borderBottomColor="white";
        rrc.style.backgroundColor="dddddd";
    }
    function reMess() {
        messItem.style.backgroundColor="azure";
        mess.style.display="none";
        messItem.style.borderWidth="0";
        rrc.style.background="red";
    }
    </script>
    <body>
        <div id="title">
        <div id="item">
            <ul>
                <li><b><font color="red">我的</font>搜狐</b></li>
                <li><img src="nobody_v_110708.gif" width="15px"/>
```

```
            wymary@sohu.com
            <img src="down.gif"/></li>
            <li>个人中心</li>
   <li id="messItem" onMouseOver="showMess()" onMouseOut="reMess()">
   消息<font id="rrc" class="rr">8</font></li>
            <li>邮件<font class="rr">99+</font></li>
            <li>博客</li>
            <li>相册</li>
            <li>说两句</li>
            <li>应用
            <img src="down.gif"/></li>
         </ul>
      </div>
<table  id="mess" onMouseOver="showMess()" onMouseOut="reMess()">
  <tr>
     <td align="center">【系统】<a href="">有 8 条系统消息</a></td>
  </tr>
  <tr style="background-color:lavender;">
     <td align="center"><a href="">查看全部消息</a></td>
  </tr>
</table>
  </div>
  </body>
  </html>
```

从这个例子中，我们能够体会到，JavaScript 的一些效果的实现，原理上其实挺简单的，在实现的过程中，对细节的控制会消耗我们大量的时间，在反复的测试过程中，也常常会蹦出来很多要处理的细节，应该接受我之前说过的 JavaScript 比 Java 还要难的说法了吧，事实上好的程序员和差的程序员之间的区别，就是这个细节，基本的功能实现叫得上程序员的都能做，可是实现的好坏就看谁对细节追求的好了，IPAD、IPhone 如此火爆，一个重要的原因就是在细节上对完美的追求。

一个好的程序员不是能够很快实现一大堆功能的人，而是能够将一个功能做到极致的人。

2.11　QQ 空间的设置

先要说明一下，在这个部分，我并不完全按照 QQ 空间的页面来实现，因为这样要消耗太多的代码，而且 QQ 空间的设置功能是为了更加实用，而我更注重功能，由此我提出一个观念，希望你能够在程序员的生涯中牢牢记住，“技术不是最重要的，最重要的是让用户舒服”，这是因为有很多技术人员会狂热的崇拜技术，而忽略用户感受，事实上存在着很多软件产品，只有专业人员才能上手，这不是真正好的产品。这个观念对于我来说根深蒂固，就像我的课或是写的书一样，我不关心教什么技术，我关心能不能让更多的人学会。

在 QQ 空间里有很多的栏目，如果用鼠标拖动，会发现一个栏目会被拖走，在 Web QQ 的网

页上也可以看到很多这样的效果，以至于 Web QQ 实现出来类似于系统桌面的效果。我们现在就来实现这个拖动栏目的效果。先用些代码准备一个栏目，就做下面如图 2-40 所示的栏目吧。

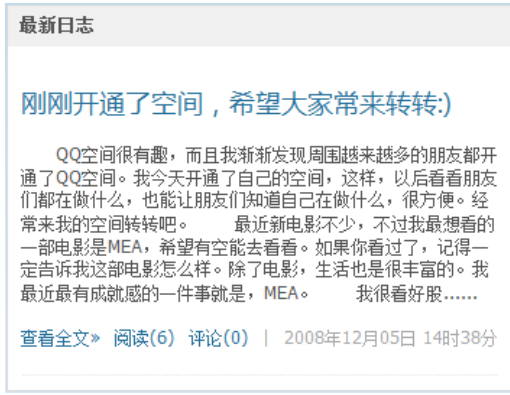


图 2-40

我们能想到，这是整个的一个 div，现在来实现吧。具体代码如下。

```
<html>
<style>
.title{
background-color:EEEEEE;
width:300px;
}
.t1{
font:normal normal 600 16px/18pts 黑体;
color:dodgerblue;
}
.context{
font-size:12px;
text-indent:25px;
line-height:20px;
}
a{
color:dodgerblue;
text-decoration:none;
font-size:13px;
}
</style>
<body>
<div id="log">
    <table style="width:350px">
    <tr>
        <td class="title">
        <div style="font-weight:bold; color:dodgerblue">最新日志</div>
        </td>
    </tr>
```

```
    <tr>
        <td>
        <div class="t1">刚刚开通了空间，希望大家常来转转:)</div><br/>
        <div class="context">QQ 空间很有趣，而且我渐渐发现周围越来越多的朋友都
开通了 QQ 空间。我今天开通了自己的空间，这样，以后看看朋友们都在做什么，也能让朋友们
知道自己在做什么，很方便。经常来我的空间转转吧。　　最近新电影不少，不过我最想看的
一部电影是 MEA，希望有空能去看看。如果你看过了，记得一定告诉我这部电影怎么样。除了
电影，生活也是很丰富的。我最近最有成就感的一件事就是，MEA。　　我很看好
股......</div><br/>
        <a href="">查看全文»</a> <a href="">阅读(6)</a> <a href="">评论
(0)</a> |
        <label style="color:BBBBBB; font-size:12px">2008 年 12 月 05 日 14
时 38 分
    </label>
        </td>
    </tr>
    </table>
</div>
</body>
</html>
```

鼠标在进入栏目的时候，颜色和内容会有些变化，这个你可以自己想办法实现。我现在想要实现的效果是在拖动的时候，栏目会跟随着鼠标移动。

内容自由移动，无疑是要设置 position:absolute;，在 JavaScript 中没有直接的鼠标拖动事件，但是有 onMouseDown、onMouseMove 和 onMouseUp，我想你应该知道如何判断鼠标拖动了吧。event.x 是鼠标的 x 坐标，当然 event.y 就是 y 坐标了，但是要强调一下，这个坐标是相对的，不一定是页面上的坐标，具体情况下需要你自己感受，在使用前，可以用 alert 将坐标值打印出来看看。

在鼠标处于标题栏的时候，鼠标的光标要变成四个方向的箭头，下面的代码先实现这个效果。HTML 部分在表格第一行的 tr 标记中使用 onMouseOver 事件以便设置光标，用 onMouseOut 事件将光标还原。

```
<tr onMouseOver="mouseOver()" onMouseOut="mouseOut()">
```

JavaScript 中的两个函数如下。

```
function mouseOver() {
    log.style.cursor="move";
}
function mouseOut() {
  log.style.cursor="default";
}
```

下面我们来处理拖动，相关的 HTML 代码是在 div 中加入鼠标的事件。

```
<div  id="log"  onMouseDown="mouseDown()"  onMouseMove="mouseMove()"
onMouseUp="mouseUp()" >
```

然后在 JavaScript 中定义 boolean 变量来确定鼠标移动的时候是否鼠标的键是按下的。JavaScript 的代码如下。

```
x = 30;
y = 30;
b = false;
function mouseDown() {
    //确定拖动的有效区域
    if(event.x>x&&event.x<(x+300)&&event.y>y&&event.y<(y+20)){
        b = true;
    }
}
function mouseMove() {
    if(b){
        //CSS 属性要提供字符串值
        x = event.x;
        y = event.y;
        log.style.left=x+"px";
        log.style.top=y+"px";
    }
}
function mouseUp() {
    b = false;
}
```

在测试的时候，我们会发现到目前为止，我的想法太单纯了，这个栏目跟随着鼠标移动没有问题，问题是在移动的一开始，栏目就立刻跑到鼠标的右下方，这带来了一个问题是 onMouseUp 事件不会发生了，因为这个事件绑定在栏目对象上，而栏目对象此时已经跑到了鼠标外边，onMouseUp 事件不起作用，变量 b 就不会变成 false，即便鼠标放开，窗体也会跟着移动。而我们想要的效果是，栏目是随着鼠标移动的，鼠标光标在栏目上的相对位置不会变化。

问题的关键是不能单纯的用鼠标坐标来定位栏目，要计算出来偏移量。修改代码如下。

```
x = 30;
y = 30;
w = 0;
h = 0;
b = false;
function mouseDown() {
    //确定拖动的有效区域
    if(event.x>x&&event.x<(x+300)&&event.y>y&&event.y<(y+20)){
        b = true;
        w = event.x-x;
        h = event.y-y;
    }
}
function mouseMove() {
    if(b){
        x = event.x-w;
        y = event.y-h;
        log.style.left=x+"px";
```

```
            log.style.top=y+"px";
        }
    }
    function mouseUp() {
        b = false;
    }
```

想必你已经能够感觉到 HTML+CSS+JavaScript 结构远比我所提供的内容博大精深，甚至超出了你的想象了吧，在软件开发的圈子里，有很多人会乐此不疲的研究这些前段技术很多年。

成为其中的高手，绝不是传统意义上的那些知识，还有思路、经验和感觉，只有不断地揣摩，不断地写代码，才能够建立这些能力，虽然这本书关于 HTML 部分的任务告了一个段落，但是你的练习还远没有结束，见到网页中好的效果，不妨尝试着自己去实现一番。在我们所完成的那些任务中，你应该也能体会到，确定如何实现的思路相比之下并不困难，困难的是实现出来，所以切忌想明白就完，一定要做出来才算。

因为 HTML、CSS 和 JavaScript 是不同的人在不同时期创建的，所以一个效果往往有多种实现方式，在特定的情况下，会有更优的实现方式，我限于时间和本书篇幅的限制，都只提供了一种实现，而且不一定是最优的实现，百度公司定义程序员的级别大体是这样的标准，从低到高排列是：掌握技术、能够解决问题、能够使用多种手段实现、能够使用全部可能的实现并能找到最优的方式、行业中没有人可以做得更好。你该知道你现在处在那个程度上了，希望你从现在去就开始尝试并思考更多解决问题的途径。

第 3 章

Servlet

3.1 Servlet 怎么运行

现在我们要实现的不是用户登录界面，而是当用户输入用户名和密码后，我们如何验证是否是合法用户，从现在开始我不会付出过多的代码在 HTML 上，而是将工作集中在后台逻辑上。

我们知道在实现用户登录的 HTML 界面的任务时，我们的工作中断在用户名和密码通过提交按钮传出去的那一刻，我们也知道用户名和密码这些信息传给了服务器，传给了 Tomcat，传给了在 form 标记，action 属性中指定的程序，这个程序就在 Tomcat 中。

这个程序不可能是 HTML，因为 HTML 没有逻辑能力，也不是 JavaScript，因为 JavaScript 只是运行在浏览器中。我们称 JavaScript 这个程序为前端脚本，有很多技术能够接受浏览器发送过来的用户信息，并且进行相应的验证，然后返回结果给浏览器。

它们都是流行的计算机语言，在 C 语言基础上衍生了 CGI（公共网关接口）编程，微软的 Web 服务器还为 C++提供了 ISAPI，随着微软大力推广.NET，ASP.NET 越来越流行，而在 Java 语言基础上，衍生出来的 Web 后台技术叫做 Servlet。

这些衍生依然遵循原有语言规范，也就是说 Servlet 程序就是百分之百的 Java 程序。

那么我们立刻就遇到一个问题，网页是从 Tomcat 那里得到的，那么从浏览器发送给服务器的信息也应该是被 Tomcat 得到的，那么我们的 Servlet 程序将如何从 Tomcat 那里拿到信息的呢，或者说是 Servlet 程序是如何同 Tomcat 配合工作的呢。

现在我们先来回顾一下 HTML 是如何来到浏览器的，如图 3-1 所示。

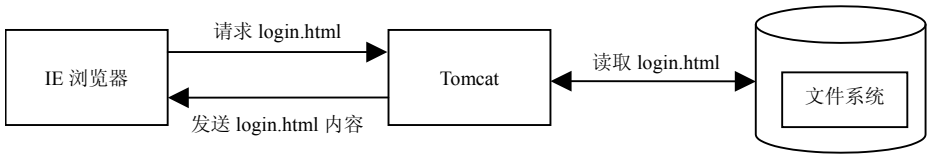


图 3-1

这个过程并不复杂，IE 浏览器告诉 Tomcat 自己需要 login.html，Tomcat 去自己管理的目录

中找来这个文件，然后将这个文件的内容通过网络发送给 IE 浏览器。

现在我们来看如果浏览器要找的是 Servlet，情况又如何呢，再次强调 Servlet 就是 Java 程序，而这个 Java 程序势必和 Tomcat 一起工作，事实上 Servlet 是通过 Tomcat 和浏览器打交道的，所以 Servlet 也处于 Tomcat 的管理之下，我们知道 Java 程序要编译成 class 文件才能真正工作，所以 Tomcat 管理的是 class 文件如图 3-2 所示。

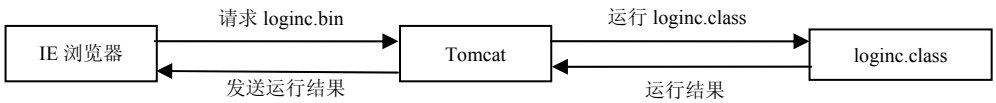


图 3-2

和访问 HTML 不同的是，访问 Servlet 是要运行 class 文件，将运行结果发送给浏览器，因为是运行结果，所以浏览器得到的内容是什么，通常在设计的时候并不确定，所以人们将通过这个过程看到的页面叫做动态页面，初学者会误解这个词，认为动态页面是有很多动态效果的页面，事实上动态页面专指网页内容是动态生成的临时网页。

我们知道一个网站可能会有很多用户同时访问，就如同在我的上一本书实现的 QQ 一样，服务器为了应对多用户的访问，会启动多线程，每个线程支持一个客户端，在 Web 环境下，Tomcat 如同 QQ 的服务器，而浏览器就是客户端了，要说两者还是有很大差异的，主要差异来源于协议的不同，但是基本原理是一致的。

一个页面可能同时会被很多浏览器访问到，如果不考虑同时，一个页面一定会多次被访问，根据上面图形对访问的描述，意味着每发生一次访问，Tomcat 就要到硬盘中找一次对应的 class 文件，下面的工作是将这个 class 变成对象，然后会调用其中的一个特定的方法，让我们写的代码运行起来，访问结束，垃圾回收机制还要清除这个对象。

如果让你来优化这个过程，如何能够提高 Servlet 的运行效率呢？Tomcat 的设计者也面临着同样的课题，人们发现如果在访问前就将 Servlet 的类创建成对象放到内存中，浏览器访问的时候，就省去了大量耗时的准备工作，只要直接调用方法运行就好了，访问结束，我们也可以不清除这个对象，等待着下次的访问。

如图 3-3 所示是 Servlet 生命周期的示意图，我们发现 Servlet 被创建了一次，使用了多次，但是这样的过程还不够精准，我们可以想象到两个问题，如果有很多的 Servlet，就要在 Tomcat 启动的时候创建很多的对象，当系统复杂到一定程度，会有一些没有用的 Servlet 存在，它们将无端的占用很多服务器资源，另一个问题是，前面提到了多线程，很多网页会被频繁访问，这样就有可能因为 Tomcat 启动了多线程，而每个线程都需要同样的 Servlet 对象，那么在 Tomcat 启动的时候，到底应该为每个 Servlet 创建几个对象呢？基于上面的讨论，现在的 Servlet 对象是在第一次使用的时候被创建的。

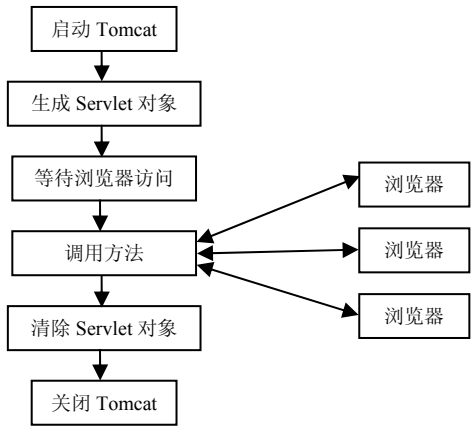


图 3-3

3.1.1 编写第一个 Servlet

现在我们将注意力放到如何编写 Servlet 上，我想你现在能理解，Servlet 就是一个 Java 的类，它是被 Tomcat 创建并调用的，那么我们就不需要再写 main 方法了，因为 Tomcat 负责写 main，而我们要写的类将依托于 Tomcat。这样我们又遇到了那个经典的问题，Tomcat 事先写好了对我要写的类的调用，那个时候我的类并没有写出来，如何确保将来调用没有问题，我们将使用接口，接口的名字是 javax.servlet.Servlet，但是你在 Eclipse 中找不到这个接口，还记得 JDBC 连接数据库的时候，我们要引入一个 jar 文件到 JDK 中吗，Java SE 的 JDK 中没有这个接口，因为 Servlet 也是 Java EE 范畴的技术，我们还要找到对应的 jar 文件来扩展 JDK，这个 jar 文件不需要我提供给你，因为它就在 Tomcat 目录的子目录 lib 中，名字叫做 servlet-api.jar，不需要将这个文件拷贝到 JDK 中，因为前面为了让 Tomcat 能够正常工作，我们设置了 java_home 这个环境变量，所以 Tomcat 自己能够通过这个环境变量找到扩展的 jar 文件。现在我们要将这个文件配置到 Eclipse 的 JDK 设置中去，做法和添加 JDBC 的 jar 文件相同，在项目的 JRE System Library 上单击鼠标右键，就会弹出右键菜单，如图 3-4 所示。

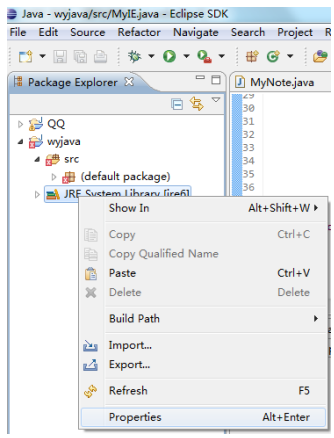


图 3-4

选择 Properties，中文版本会翻译成“首选项”，默认弹出的便是配置 JDK 的对话框，如图 3-5 所示。

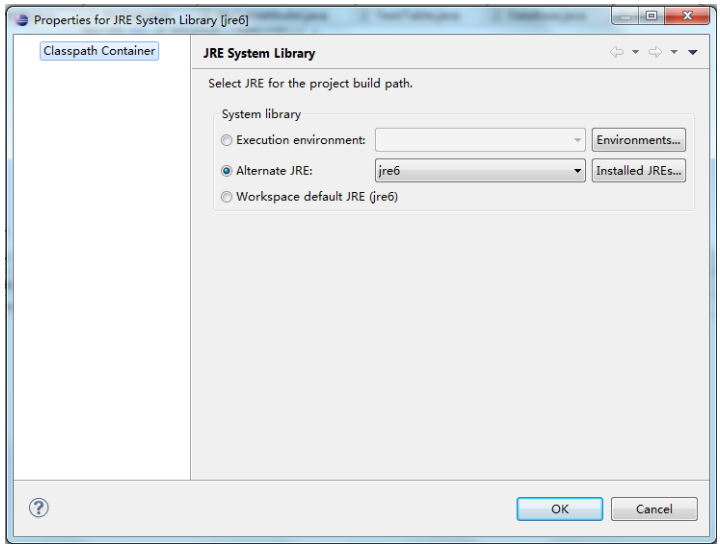


图 3-5

单击 Installed JREs 按钮选择添加的 jar 文件，如图 3-6 所示。

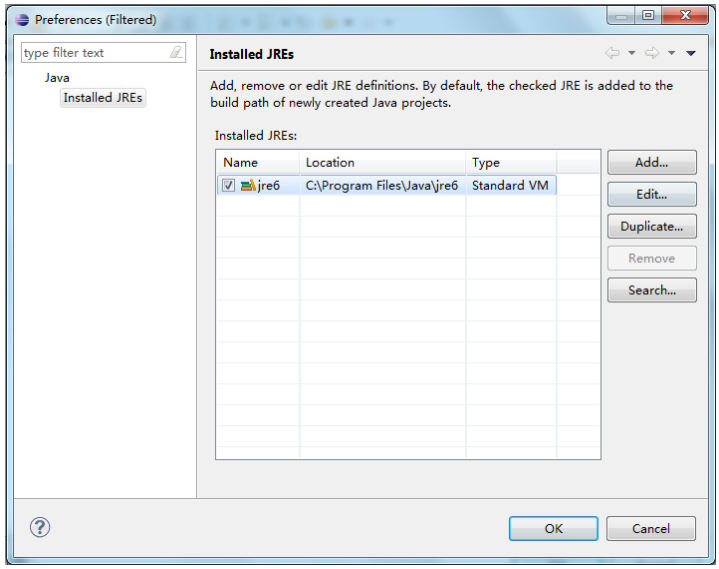


图 3-6

用鼠标选中列表中显示出来的 jre，Edit 按钮会变黑，然后单击 Edit 按钮，弹出如图 3-7 所示的 Edit JRE 对话框。

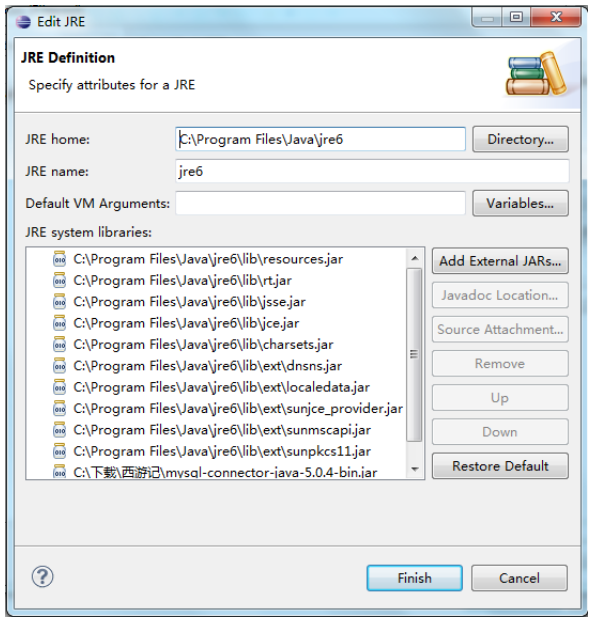


图 3-7

单击添加扩展的 jar（Add External JARs）按钮，找到 Tomcat 目录里面的 lib 目录，选择 servlet-api.jar 文件，到此配置 JDK 的工作就完成了。

大多数讲授使用 Eclipse 开发 Java Web 应用的书，现在该教你如何将 Tomcat 集成到 Eclipse 开发环境中了，这样做的好处是，不离开 Eclipse，你就可以进行 Servlet 的部署，并且实现对 Tomcat 的控制，而我现在并不提供这样的指导，因为我只是将 Eclipse 当成相对方便的 Java 编写和编译环境，其他的丰富功能我并不想使用，只有这样你才能清楚自己一步步在做什么，如果要 Eclipse 代替你来做这些事情，你很可能写出了代码，却不知道为什么。

现在我们来写一个最简单的 Servlet 来学习如何开发和部署，我们就在浏览器上打印一句话：“我的第一个 Servlet”。

和 Java 程序一样，我们要新建一个类，将其命名为 FirstServlet，这次我准备了包 com.wy，现在在生成的代码上面实现接口 javax.servlet.Servlet，过去我们已经进行过多次，需要 Eclipse 协助你将所有抽象的方法都实现出来，最终我们得到的代码如下。

```
package com.wy;

import java.io.IOException;
import javax.servlet.*;

public class FirstServlet implements Servlet {

    @Override
    public void destroy() {
        // TODO Auto-generated method stub
```

```
    }

    @Override
    public ServletConfig getServletConfig() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public String getServletInfo() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public void init(ServletConfig arg0) throws ServletException {
        // TODO Auto-generated method stub

    }

    @Override
    public void service(ServletRequest arg0, ServletResponse arg1)
            throws ServletException, IOException {
        // TODO Auto-generated method stub

    }
}
```

现在我们来看一下实现的这 5 个方法，在讲解之前我描述 Servlet 生命周期的特点，在这个 Servlet 第一次被访问的时候，Tomcat 会创建它的对象，然后这个对象会经历很多次的调用，当 Tomcat 关闭的时候，这个对象才被清除掉，也就是说在 Servlet 的生命周期里，一个 Servlet 会被创建一次，清除一次，而会被调用很多次。

init 方法便是在开始的仅有的一次创建时被 Tomcat 调用，通常我们在这里做一些初始化的工作，也可以什么都不做。对应的 destroy 方法是在该对象被清除的时候，会被 Tomcat 调用一次。

getServletInfo 方法和 getServletConfig 方法事实上跟 Servlet 生命周期没有什么关系。getServletInfo 提供的是这个 Servlet 的信息，有时我会在这里提供程序员名字，还可能是版本号之类的信息，当然大多数情况下，这个方法什么都不做，空在那里。getServletConfig 方法相对来说要重要多了，我们知道一个类通常完成了一段特定的功能，或者成为逻辑，我们有时希望写好的一个类能够有更加强大的适应能力，可以覆盖相似的逻辑，有两个办法让类更加强大，一是通过带参数的构造方法，可以利用参数改变类的功能，另一个办法是使用配置文件，问题　　是 Servlet 并不是一个独立的程序，它存在于 Servlet 引擎中，在我们这里 Servlet 引擎就是 Tomcat，我们写好的 Servlet 将成为 Tomcat 的一个部分，读取配置文件的工作 Tomcat 已经帮助我们做

了，我们写的 Servlet 需要向 Tomcat 要配置信息，这个工作就由 getServletConfig 方法来完成，返回值是 servletConfig 对象。

剩下最后一个方法，也是最重要的方法，service(ServletRequest arg0, ServletResponse arg1)，这就是每次有浏览器过来访问，Tomcat 要调用的方法，在 Servlet 的生命周期中，service 可能会被调用很多次。

我们看到 service 方法中有两个参数，它们都是对象，第一个 ServletRequest 对象包含了从浏览器传送过来的一切，其实浏览器能传送过来的不过是地址栏里面的字符串（还可能包含隐含传送的信息），字符串事实上是被传送到 Tomcat，Tomcat 接收以后将字符串按照协议进行了初步的处理，包装成为 ServletRequest 对象，在调用 service 方法的时候，作为第一个参数传给你的程序。

第二个参数是指向浏览器的对象，第二个参数和第一个参数是反向的，如果我们要输出信息到浏览器，就要通过 ServletResponse 对象，这个对象包含 Tomcat 指向浏览器的一切资源，最重要的就是 Tomcat 的输出流。

这样说来，我们要显示一句话到浏览器上，就需要第二个参数 ServletResponse。

我们看到在 service 方法的后面有这么一段语句，**throws** ServletException, IOException。还记得异常处理吗？当异常发生的时候，程序会落入 catch 中，我们可能会写程序在 catch 中处理异常，throws 是另一种做法，将 ServletException 和 IOException 异常抛出去，这是个看上去并不负责任的做法，不去处理异常，而是将异常丢出去，那么丢给了谁呢，谁调用这个方法，就会丢给谁，在我们的程序中，是 Tomcat 调用 service 方法，所以一旦出现了这两个异常，异常就会丢给 Tomcat。我们完全可以自己来处理这两个异常，其实这么做也没什么不可以的，比如，IOException 就是应该由 Tomcat 来处理，因为毕竟 IO 流是 Tomcat 创建的，我们的程序只是使用 IO 流，根本就没有能力处理由此产生的异常。

现在我们在 service 方法中填上一句话。

```
arg1.getWriter().println("我的第一个 Servlet");
```

我们的程序代码就算是写完了，存盘就完成了编译工作，现在我需要生成的 class 文件。

3.1.2 部署

因为使用了强大的 Eclipse，找到 class 文件还真不容易，还记得你设置的工作空间是什么吗？如果不记得了，就在工程的名字上单击鼠标右键，在右键菜单中，找 Properties 首选项。

如图 3-8 所示，第一项 Resource 中，有相关信息，比如，我当时的设置，Location 后面的内容，c:\myjava\wyjava，现在就去那里找 class 文件吧。class 文件被放在再下一层的 bin 中，因为有包存在，所以在这里你能看到一个子目录 com，进去是子目录 wy，再进去才是我们的类 FirsrServlet，这是编译有包的文件所产生的效果，在使用的时候，子目录结构是类的一个部分，要一起使用。

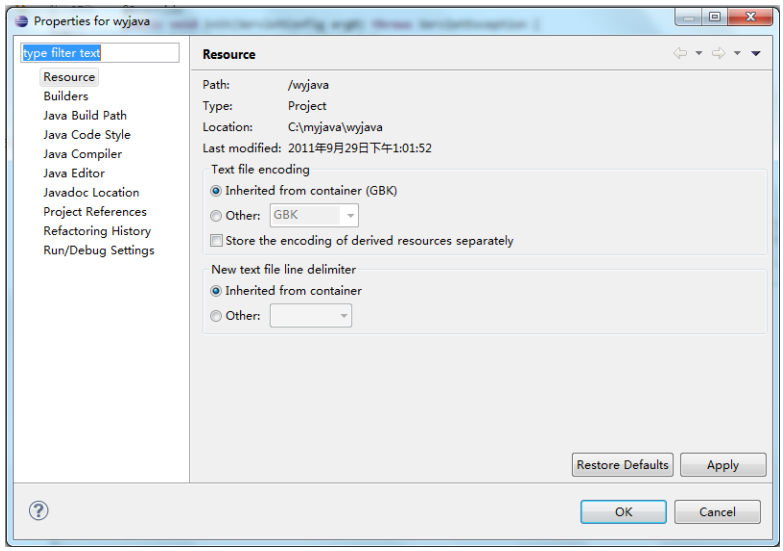


图 3-8

类写好了，也编译完了，怎么让它工作起来，我不止一次的提到，Servlet 是 Tomcat 系统的一个部分，至少我们要将这个 Servlet 类文件复制到 Tomcat 中。

还记得之前 HTML 放在什么地方吗，它在 webapps\ROOT 中，这个 ROOT 目录是默认的 Tomcat 的根目录，问题是未来我们不能将所有的项目都放到 ROOT 里，看样子我们需要一个独立的目录，其实在 webapps 中已经有好几个子目录了，这每一个都是 Tomcat 管理下的一个目录，现在我们在 webapps 里面创建一个我们自己的目录 study。

这样的一个空目录 Tomcat 是不认的，在 study 中还需要一个叫做 WEB-INF 的子目录，这个子目录中需要一个 web.xml 的文件，建议你将 ROOT 目录中对应位置的 web.xml 文件复制过来。如图 3-9 所示。

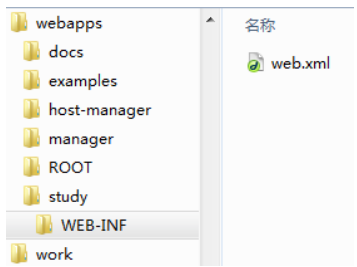


图 3-9

这时 study 这个目录才被 Tomcat 接受，现在在 study 这一级目录下随便写个 html 文件，然后启动 Tomcat，用浏览器访问 http://127.0.0.1/study/test.html，如果没有问题我们继续。

在 WEB-INF 中创建子目录 classes，然后将编译好的 class 文件连同包目录一起复制到 classes 中。classes 目录不是必须的，但是这个名字是规定好的，如图 3-10 所示。

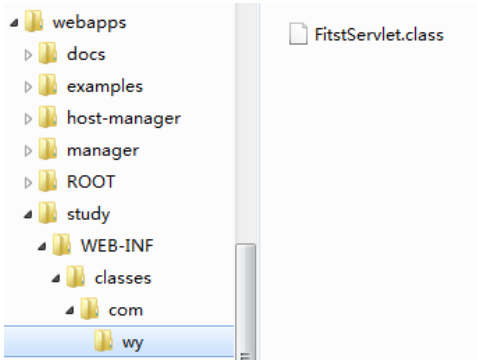


图 3-10

现在我们编写的 Servlet 已经被放到 Tomcat 中了，但是 Tomcat 还是不知道这个 Servlet 的存在，需要改写配置文件，将复制过来的 Servlet 添加进去，以便 Tomcat 在启动的时候，能够找到并创建这个 Servlet 对象。

配置文件就是刚才从 ROOT 目录复制过来的 web.xml，为什么用 xml 来做配置文件，以及 xml 是怎么回事，我随后专门讨论，现在我们的目标是用起来就好，打开这个文件来看看。文件内容如下。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  version="3.0"
  metadata-complete="true">

  <display-name>Welcome to Tomcat</display-name>
  <description>
     Welcome to Tomcat
  </description>

</web-app>
```

其中主要的内容是说明，它们被放在<!-- -->中注释着，为了节约篇幅，我将这部分内容删除掉了。

我先将写好的 Servlet 配置上去以后，再来分析这个 XML，添加如下代码。

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<web-app xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  version="3.0"
```

```
metadata-complete="true">

<display-name>Welcome to Tomcat</display-name>
<description>
   Welcome to Tomcat
</description>

<servlet>
  <servlet-name>first</servlet-name>
  <servlet-class>com.wy.FirstServlet</servlet-class>
</servlet>

 <servlet-mapping>
  <servlet-name>first</servlet-name>
  <url-pattern>/first.bin</url-pattern>
 </servlet-mapping>

</web-app>
```

总体看起来是不是很像 HTML，没错，这也是标记语言，事实上 HTML 是 XML 的一个分支，只不过在演变的过程中，由于浏览器的宽松，所以 HTML 的语法变得不太严谨了，而 XML 依然保持着严谨的语法。

第一行<?xml version="1.0" encoding="ISO-8859-1">，这是 XML 文件的第一行，是必须的，包括版本信息也是必须的，encoding 要根据文件的内容进行修改，如果有中文的话就不是 ISO-8859-1，而是 GB2312 了。

下面的内容全部被放到<web-app>标记中，这有点像 HTML 的<html>，区别是在 HTML 中<html>标记是可以省略的，而这个文件中<web-app>不能省略，因为 XML 有个单根结构的规定，你看到整个文件的标记是不是一层层深入的，事实上这是个树形结构，我们用图形形象的将这个 web.xml 文件的节点描述出来，就好理解了，如图 3-11 所示。

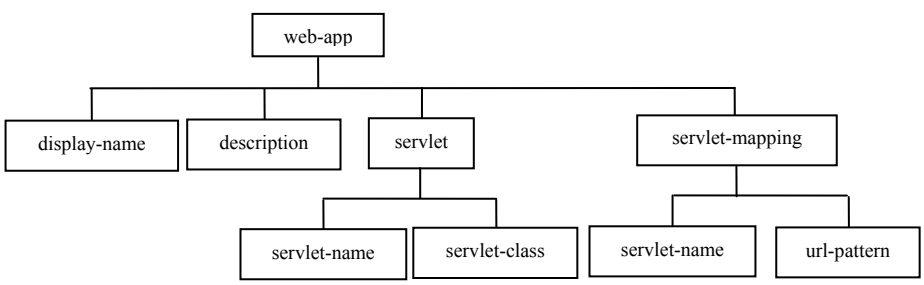


图 3-11

这就是上面 XML 文件的树形结构图，能看出来这个树形结构的根是 web-app，一个合法的 XML 中只能有一个根节点，如果违反了这个规定，文件就不是 XML 文件了。

你是否意识到这个文件里面的标记不是 HTML 标记，完全是另一套系统，没错，这是另外

定义的，和 HTML 标记的定义不同，HTML 标记是每个人都知道的规则，而 web.xml 中能够使用的标记是用另外的文件定义的，那个文件被说明在<web-app>标记的 xmlns 属性中，这个我们回头再细说。

下面的两个标记应该能够猜出来，只不过是名字和说明。现在来看我添加的标记，一共有两个，一个是 servlet，一个是 servlet-mapping，servlet 中的第二个标记叫做 servlet-class，看里面的内容不难理解，就是我们写的那个 Servlet 的类，上面的 servlet-name 是在 Tomcat 中这个类所生成对象的名字。

我们知道不管是什么理由，web.xml 是 Tomcat 的配置文件，也就是说 Tomcat 会读这个文件的内容，我们将写好的 Servlet 放到这个文件中的目的是什么呢，就是让 Tomcat 能够找到这个类文件，并且创建对象以便浏览器访问的时候调用。换一个角度，servlet 标记产生的效果是，Tomcat 读这个标记的内容，然后将类创建成对象，对象的引用名字就是 servlet-name。我们这个例子，如果翻译成 Java 代码是，com.wy.FirstServlet first = new com.wy.FirstServlet() ;，Tomcat 不仅仅创建了对象，还会调用 first.init()。

但是 Tomcat 一定不是这样实现代码的，因为 Tomcat 只能从 web.xml 文件中读到字符串，必须使用字符串创建对象。我们学过上一本书中的反射，如果不清楚反射的话，你要找到那本书中的反射内容，复习一下，通过这个过程，理解如果是你编写的 Tomcat，你会怎么做。

servlet-mapping 的作用是提供用户访问的 url，servlet-name 和 servlet 标记中的名字一样，这样 url 才能和具体的类关联到一起。url-pattern 提供的就是访问时地址栏上的 url 信息，注意前面有个斜线/，扩展名可以是任意的字符串或是没有扩展名，.bin 是约定俗成的，用来表示这是一个程序，而不是静态网页。

编写好后将 web.xml 存盘，看看我们的成果吧，启动 Tomcat，如果你的 Tomcat 已经运行了，一定要关掉它，然后再重新启动，只有这样，新加入的 Servlet 才能被 Tomcat 调用到。然后用浏览器访问 http://127.0.0.1:8080/study/first.bin，就能看见我们要打印的那句话了。

3.2 用户登录

如果可以的话，我们来完成实用的功能。针对用户输入的登录信息进行验证，我们先准备 login.html 文件，这个文件放在 study 这一级的目录下，只有类文件放在 classes 中。

为了节约篇幅我提供的是没有任何美化的用户登录界面，下面是 login.html 的文件内容。

```
<html>
    <body>
        <form action="loginc.bin">
            用户名：<input type="text" name="username"/><br/>
            密码：<input type="password" name="password"/><br/>
            <input type="submit" value="登录"/>
        </form>
```

```
    </body>
</html>
```

请注意看 action 属性的内容，现在知道 loginc.bin 是什么了吧。我们首先启动 Tomcat，然后用浏览器访问 http://127.0.0.1:8080/study/login.html，应该能够看到这个用户登录的网页，在用户名和密码的输入框中输入信息，然后单击登录按钮，因为 loginc.bin 还没有编写，所以看到的将是如图 3-12 所示的页面。

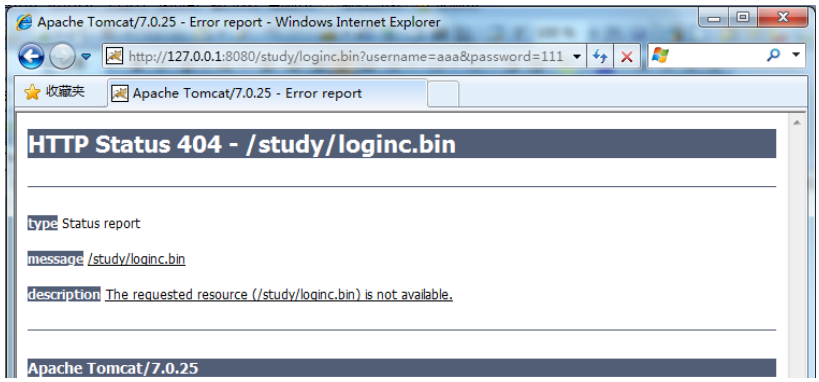


图 3-12

我们得到了一个 404 错误，404 错误表示请求的网页不存在，你仔细看一下浏览器的地址栏，能猜出来我输入的用户名和密码是什么吗？我们看到 loginc.bin 后面还有一些内容，?username =aaa&password=111，这是浏览器传送到服务器最主要的信息，在设计之初，浏览器和服务器之间的传输就是不平衡的，浏览器通常只会将相对简单的地址栏信息发送给服务器，而服务器通常会将复杂的网页传送回来，所以上行的带宽需求较低，下行需求较高。

而这句话?username=aaa&password=111 被 Tomcat 接收后，便会用？来识别，用&来分隔，将信息包装成 Request 对象，以参数的形式传递到 Servlet 中。

问题是，你是否意识到，这个网页不安全，如果在网吧里上网，你是否会担心身边可能站着人，或是头顶上会有摄像头，虽然在输入的时候 password 输入框不会显示具体的字母，但是地址栏中有相关内容的显示。

如果希望地址栏中不显示具体信息，就要设置传送方式为 POST，默认的情况下是 GET，还记得我们做假 Tomcat 的时候，第一行接收的第一个单词就是 GET，指的就是默认的 GET 传送方式，通过 form 标记的 method 属性将 GET 方式修改成 POST 方式。<form action="loginc.bin" **method="POST"**>，要注意的是，GET 和 POST 的设置都是大写字母，虽然 HTML 对大小写要求不高，但是由于这个地方用来设置 HTTP 协议头，所以要遵守 HTTP 协议的规则，一旦不符合 HTTP 协议的规则，我们的浏览器会认为这不是一个网页，现在的计算机允许浏览器访问非网页，只是大多数情况下浏览器会认为非网页是文件，默认处理文件的方式是下载，如果你的计算机没有安装下载工具，浏览器就会使用自带的文件下载功能，如果安装了下载工具，下载工具程序就会被启动，包括以后，如果发现测试的时候出现了下载的界面，通常先考虑是不是 HTTP 协议设

置错了。另外这个修改只会影响这次传输，其他网页的传输方式还是 GET 方式。

修改一下 HTML 文件，看看是不是地址栏中不显示用户名和密码了，这样做还有一个好处，浏览器的地址栏本身是有字数限制的，如果要传输的内容太多，使用 Get 方式，会截断超范围的字符。

现在我们来编写 loginc.bin，先来写类 Loginc。具体代码如下。

```
package com.wy;

import java.io.IOException;
import javax.servlet.*;

public class Loginc implements Servlet {
    @Override
    public void destroy() {
    }

    @Override
    public ServletConfig getServletConfig() {
        return null;
    }

    @Override
    public String getServletInfo() {
        return null;
    }

    @Override
    public void init(ServletConfig arg0) throws ServletException {
    }

    @Override
    public void service(ServletRequest arg0, ServletResponse arg1)
            throws ServletException, IOException {
        String user = arg0.getParameter("username");
        String pass = arg0.getParameter("password");
        if(user.equals("aaa")&&pass.equals("111")){
            arg1.getWriter().println("欢迎");
        }else {
            arg1.getWriter().println("验证失败");
        }
    }
}
```

重点在 service 方法中，之前提过 ServletRequest 负责存放浏览器发送过来的信息，所以用 ServletRequest 对象的 getParameter 方法获取用户的输入，参数 username 和 password 想必你能猜到，就是表单输入框的名字。输出欢迎和验证失败在上个案例中已经这样应用过了。

现在我们要部署这个 Servlet，将产生的类文件连同生成的包目录一起复制到 study\Web-INF\classes 中。然后改写 web.xml，添加新的 Servlet 描述。具体代码如下。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  version="3.0"  metadata-complete="true">

  <display-name>Welcome to Tomcat</display-name>
  <description>
     Welcome to Tomcat
  </description>
  <servlet>
    <servlet-name>first</servlet-name>
    <servlet-class>com.wy.FirstServlet</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>loginc</servlet-name>
    <servlet-class>com.wy.Loginc</servlet-class>
  </servlet>

   <servlet-mapping>
    <servlet-name>first</servlet-name>
      <url-pattern>/first.bin</url-pattern>
  </servlet-mapping>
   <servlet-mapping>
    <servlet-name>loginc</servlet-name>
      <url-pattern>/loginc.bin</url-pattern>
  </servlet-mapping>

</web-app>
```

部署妥当后，重新启动 Tomcat，我们再用浏览器进行一下测试。如果输入了 aaa 和 111，我们并没有得到“欢迎”的字样，而是两个??，没错这就是欢迎，只是 Tomcat 默认的情况下传输的不是中文，当然有些版本的 Tomcat 会默认传输中文。看来我们要用程序设置传输的语言了。下面我只提供 service 方法。具体代码如下。

```
public void service(ServletRequest arg0, ServletResponse arg1)
    throws ServletException, IOException {
String user = arg0.getParameter("username");
String pass = arg0.getParameter("password");
arg1.setCharacterEncoding("GB2312");
if(user.equals("aaa")&&pass.equals("111")){
    arg1.getWriter().println("欢迎");
}else {
    arg1.getWriter().println("验证失败");
```

```
    }
}
```

事实上也有人这样设置传输的字符集。

```
arg1.setContentType("text/html;charset=GB2312");
```

这样设置直接修改的是 HTTP 协议头，而上一种做法程序会帮你转换成 HTTP 协议头需要的格式，要注意使用 setContentType 时，字符串要严格按照我的实例来写，分号前后也不能加入空格，否则就会出现文件下载页面，这是 HTTP 协议的要求。

你修改了程序吗？请不要直接运行，别忘了还要部署，要将新生成的 class 文件复制到 Tomcat 中，要重启 Tomcat，只有这样才能让新的代码随着 Tomcat 启动。除非，可以设置 Tomcat，让更新的 Servlet 直接运行起来，这样开发起来就要舒服很多，另外，在 Eclipse 中集成 Tomcat 会让很多工作都自动完成，如果想省事的话，可以自己琢磨着如何设置，到网上搜索是个不错的主意。

有一件事情有点郁闷，大多数情况下，这 5 个方法中只有 service 会经常被用到，但是由于接口的规则，其他的 4 个方法还要实现一遍，有没有办法只写 service 方法。我想到了一个比较可行的做法，先写一个类 WyServlet，这个类的具体代码如下。

```
package com.wy;

import java.io.IOException;
import javax.servlet.*;

public class WyServlet implements Servlet{
    @Override
    public void destroy() {
        // TODO Auto-generated method stub
    }

    @Override
    public ServletConfig getServletConfig() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public String getServletInfo() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public void init(ServletConfig arg0) throws ServletException {
        // TODO Auto-generated method stub

    }
```

```
    @Override
    public void service(ServletRequest arg0, ServletResponse arg1)
            throws ServletException, IOException {
        // TODO Auto-generated method stub
    }
}
```

类 WyServlet 只是对接口 Servlet 进行了空的实现，这样我们再写 Servlet，只要继承 WyServlet 就好了。Loginc 的具体代码变成如下形式。

```
package com.wy;

import java.io.IOException;
import javax.servlet.*;

public class Loginc extends WyServlet {
    @Override
    public void service(ServletRequest arg0, ServletResponse arg1)
            throws ServletException, IOException {
        String user = arg0.getParameter("username");
        String pass = arg0.getParameter("password");
        arg1.setContentType("text/html;charset=GB2312");
        if(user.equals("aaa")&&pass.equals("111")){
            arg1.getWriter().println("欢迎");
        }else {
            arg1.getWriter().println("验证失败");
        }

    }
}
```

现在是否理解这个做法的原理了？WyServlet 已经实现了所有的方法，Loginc 只不过覆盖了 service 方法，如果需要的话，你完全可以覆盖其他方法。不仅仅我有这个聪明的想法，Tomcat 也想到了，它提供的空实现的 Servlet 叫做 javax.servlet.GenericServlet。

下面我们要在这个基础上到数据库中验证用户登录，我想你可以先自己尝试这样做，这并不是新的技术，只是将 Servlet 和 JDBC 结合起来而已。具体代码如下。

```
package com.wy;

import javax.servlet.*;
import java.sql.*;

public class Loginc extends GenericServlet {
    public void service(ServletRequest arg0, ServletResponse arg1) {
        try {
            arg1.setContentType("text/html;charset=GB2312");

            String user = arg0.getParameter("username");
```

```
            String pass = arg0.getParameter("password");

            Class.forName("org.gjt.mm.mysql.Driver");
            Connection cn = DriverManager.getConnection(
                  "jdbc:mysql://127.0.0.1:3306/qq", "root", "123456");
            PreparedStatement ps = cn.prepareStatement("select * from
user where username=? and password=?");
            ps.setString(1, user);
            ps.setString(2, pass);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
               arg1.getWriter().println("欢迎");
            } else {
               arg1.getWriter().println("验证失败");
            }
         } catch (Exception e) {}
      }
}
```

这里隐藏着一个问题，如果验证通过显示“欢迎”，否则显示“验证失败”，但是在现实的项目中，不可能就显示一个“欢迎”，通常会显示一个非常复杂的网页，这就意味着在验证成功的 if 语句中要输出大量的 HTML 代码，而失败也要输出大量的代码，这样实现的网站会带来严重的管理问题，因为很多功能揽到了一起，甚至很多网页都可能揽到一起。

解决这个问题的思路是，放一个跳转语句在显示“欢迎”或“验证失败”的位置，将程序的流程引导到一个独立的 HTML 文件或 Servlet 程序中。

有两个途径可以让网页跳转，一是发送一条 JavaScript 的跳转代码到浏览器，浏览器读到这个跳转代码，会重新向服务器请求新的网页。另一个方式是 HTTP 协议本身提供了跳转的指令，通过 HTTP 协议进行跳转。

第一种方式的实现是将 arg1.getWriter().println("欢迎");这句话替换成 arg1.getWriter().println("<script>location.replace('welcome.bin');</script>");，测试的结果是找不到 welcome.bin 的 404 错误，这就是正确结果，因为我们还没有提供 welcome.bin。

第二种方式用实现 Servlet 接口的方式无法实现，因为这种方式并不直接支持 HTTP 协议，可以通过继承另一个类来实现 HTTP 跳转，类名是 javax.servlet.http.HttpServlet。使用 HttpServlet 实现上面相同功能的代码如下。

```
package com.wy;

import javax.servlet.http.*;
import java.sql.*;

public class Loginc extends HttpServlet {
   public void doPost(
         HttpServletRequest arg0, HttpServletResponse arg1) {
      try {
```

```
            arg1.setContentType("text/html;charset=GB2312");

            String user = arg0.getParameter("username");
            String pass = arg0.getParameter("password");

            Class.forName("org.gjt.mm.mysql.Driver");
            Connection cn = DriverManager.getConnection(
                  "jdbc:mysql://127.0.0.1:3306/qq", "root", "123456");
            PreparedStatement ps = cn.prepareStatement("select * from
user where username=? and password=?");
            ps.setString(1, user);
            ps.setString(2, pass);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
               arg1.sendRedirect("welcome.bin");
            } else {
               arg1.getWriter().println("验证失败");
            }
         } catch (Exception e) {}
      }
}
```

不同之处我都加粗显示出来了，看到方法的名字叫做 doPost，能够猜到如果接收 Get 方式发生过来的信息，方法应该叫什么。

现在给你一个任务，如果验证失败，跳转到注册页面 reg.html，编写这个页面，并且编写 regc.bin 程序，以便将用户输入的信息添加到数据库中。

3.3　重要的 XML

能够看出我有多重视 XML，这是第一段没有多少明确代码的大章节。我们知道现在的计算机行业是由软件企业推动的，不同于过去被科学家推动的年代，受到商业利益的影响，现在的技术形成了很多门派，一方面这样的局面使得技术发展非常丰富繁荣，可是也存在流行的技术往往不是最先进的技术，因为市场地位干扰了技术发展。

Java 几乎和 C#水火不容，其他方面也是如此，好在国际标准化组织的努力工作，在很多相互竞争的技术中间，建立了标准，否则程序员可就太痛苦了，设想一下，如果 Java 程序员为了拥有使用数据库的能力，要学习每一种流行的数据库产品，会让人多么恼火。

即便是有国际标准，但是计算机行业的主体还是被商业集团分隔的，这中间只有一个例外，那就是 XML，一个被全世界所有技术厂商都共同认同的标准。

XML 如此特别，源于它所能做的事情，我们已经非常清楚 XML 是可以做配置文件的，相比于其他配置文件的规范，XML 的好处在于通过自定义的标记，程序利用标记寻找相应的配置信息，这样就可以随意调整配置信息的位置，标记也能帮助用户理解配置信息的含义。

我想你不会反对 XML 文件可以存放数据，事实上任何文件都可以存放数据，而 XML 不仅仅能够存放数据，标记的运用也管理了数据，这就如同数据库软件，在存放数据的基础上，还能对数据进行组织和管理才真正成为数据库。

当今的数据库都是使用二维表存放数据的，前面提到的 XML 是树形结构，这在数据库理论研究中进行过分析，二维表的优势在于访问和管理比较方便，但是树形结构更加贴近现实社会结构。

所以说 XML 完全可以担当数据库的职责，并且还有它特有的优势，现在的关系型数据库管理系统通常从性能上考虑太多，以至于系统相当的庞大，相当的占用计算机资源，但是在很多场合下，我们希望数据库是轻量级的，可以方便移动，否则就得为了给用户介绍产品，不得不在笔记本电脑上安装 Oracle 了。XML 作为文本文件，足够的轻量级。

为什么不直接用文本文件来存放数据呢？和纯粹的文本文件不同的是 XML 拥有标记，也发展出来很多配套的工具，最重要的是 XML 文件的标准被整个计算机产业界支持，所以 XML 不单单用来做轻量级的数据库，而且还用于在不同数据库之间迁移数据。

我们常常会遇到不同的商业伙伴之间会有密切的业务往来，彼此之间的信息系统需要对接，但是有可能双方使用的是不同的数据库产品，虽然这些数据库通常都支持着 SQL-92 标准，但是经过演变，各种数据库或多或少的发展了自己的方言，这样在不同的数据库之间交换数据，变成了令人头疼的问题。比如，我们通过中国移动打手机，而常常要到银行交话费，或许银行用的数据库是 DB2，而中国移动用的是 Oracle。还有一种情况是企业兼并，也会遇到不同的数据库融合的问题。DB2 不会开发将数据迁移到 Oracle 的工具，Oracle 也不会这么做，毕竟是直接竞争对手，现在有了 XML，问题就容易解决了，因为 XML 被十分广泛的支持，所以 DB2 愿意开发将数据迁移到 XML 和从 XML 读取数据的工具，而 Oracle 也愿意这样做，加上 XML 本身是国际标准，所以 XML 由此成为不同数据库之间的数据迁移工具。

除了在数据库领域 XML 利用自身标准帮助迁移数据，在非数据库领域，XML 也能用来协调不同计算机语言，不同服务器平台等这些场合的数据融合。

我不再一一列举，随着我们眼界的扩大，你自己会发现 XML 的更多应用领域，所以 XML 是当今软件技术的重要基础。但是我们能够看到的单纯 XML 文件应对如此多的应用还是力不从心，好在配套 XML 技术还有很多的分支可以解决不同的问题，我并不尝试着成为大而全的 XML 词典，对 XML 的讨论还是基于有用，能掌握的原则。下面我将讨论 XML、DTD、Schema、CSS、XSL、DOM 和 SAX，是不是光听名字就头疼了，虽然这样，我还是强烈建议你能够认真地面对这个特殊的文本文件，因为它是你未来学习的基石。

3.3.1 XML

因为在前面使用 web.xml 的时候，我已经对 XML 的规则进行了相当程度的阐述，所以这里只是做些总结。

XML 文件要以.xml 扩展名结尾，事实上很多场合下并不使用这个扩展名，以后你一定能够

碰到。

XML 文件的第一行是<?xml version="1.0"?>，encoding 不是必须的。前面的？和 XML 之间不能有空格，XML 的语法规定是相当严格的。

XML 的标记以<>为开头</>为结尾，完整的这样一个结构被称为元素，和 HTML 不同的是 HTML 的标记是事先规定好的，而 XML 的标记是作者自己规定的。

XML 必须是单根结构。

XML 解释器，虽然记事本这样的工具能够打开 XML 文件，但是只是将它作为文本文件来看，有些工具能够分析 XML 文件是否合法，并且能够在一定程度上理解 XML 的标记范围，这样的工具是 XML 解释器，最常见的 XML 解释器是浏览器，现在你可以试着用浏览器打开 web.xml，看到的效果如图 3-13 所示。

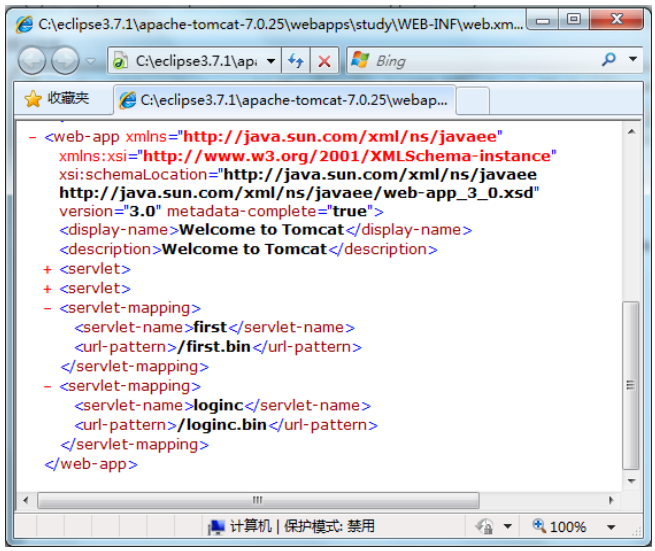


图 3-13

你会发现，甚至可以将一些标记折叠起来，那么你再试试，如果这个 XML 文件有错误，比如，破坏它的单根性，然后用浏览器打开看看效果。

浏览器只能解释 XML，我们需要一个能够编辑 XML 的工具，当然记事本可以，但是，如果有些提示功能，我们就可以更轻松的探索了，事实上 Eclipse 是编辑 XML 的工具，但是，有一个叫做 XMLSpy 的工具更加强大，XMLSpy 是收费软件，不过它也提供免费的版本。官网地址是 www.altova.com，官网上就有中文版本可以下载，安装后你可以输入一个 XML 文件体验一下。

3.3.2　DTD

前面提到 XML 的标记不是语言事先规定的，作者可以自己来定义标记，但是，这样也带来

了一些问题，在使用的过程中，别人如何理解自定义的标记，更主要的是在别人修改的时候，如何能够遵循原始作者的定义。

从另一个角度来看，我们将 XML 有时作为轻量级的数据库使用，回顾一下，数据库的操作事实上存在两个阶段，数据定义阶段和数据操作阶段，在数据操作前，通常要定义表结构，以便使数据可以按照设定有条不紊的存放。

那么，DTD 就是数据类型定义（Document Type Definition）文件，DTD 文件用于说明特定 XML 文件的规则。

我们先来创建一个 XML 文件，将其命名为 School.xml。和大多数软件一样，可以在工具栏中找到最左边新建 XML 文件的按钮，单击后，会弹出一个对话框，让你选择创建什么类型的文件，从这个对话框中能够体会到 XML 广泛的应用领域。如图 3-14 所示。

图 3-14

我们选择扩展的标记语言(Extensible Markup Language)，单击确定按钮后会出现第二个对话框，这个对话框询问你，根据什么来创建 XML 文件。如图 3-15 所示。

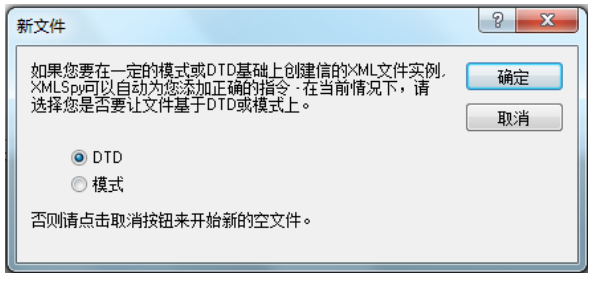


图 3-15

前面讲过 DTD 文件是在 XML 的规则之下对特定 XML 文件进一步约束的工具，这里便问你要使用那个 DTD 文件进行约束，下面的模式英文名字叫做 Schema，我会在加一个单元进行讨论，现在我们没有 DTD 或是 Schema，单击取消按钮，这样就可以编写没有标记约束的 XML 文

件了。在文件中输入下面的代码。

```
<?xml version="1.0" encoding="UTF-8"?>
<学校>
    <学生 学号="001">
        <姓名>张三</姓名>
        <性别>男</性别>
        <年龄>20</年龄>
        <成绩>
            <Java>85</Java>
            <SQL>90</SQL>
            <HTML>79</HTML>
        </成绩>
    </学生>
    <学生 学号="002">
        <姓名>李四</姓名>
        <性别>男</性别>
        <年龄>19</年龄>
        <成绩>
            <Java>87</Java>
            <SQL>76</SQL>
            <HTML>90</HTML>
        </成绩>
    </学生>
    <学生 学号="003">
        <姓名>王五</姓名>
        <性别>男</性别>
        <年龄>21</年龄>
        <成绩>
            <Java>73</Java>
            <SQL>82</SQL>
            <HTML>45</HTML>
        </成绩>
    </学生>
</学校>
```

事实上学校还会有其他信息，我们现在先不考虑这么复杂，请一定要按照我的内容输入，在输入的过程中体会 XMPSpy 给你提供的帮助，要注意的是，我特地使用了中文标记，说明可以随意的设置标记，但是有些中文输入法中<和>于英文的不同，这样会产生错误，输入的时候要注意这点。

现在我们就这个 XML 文件来编写 DTD，你要创建新文件，在文件类型选择的对话框中找 DTD，对话框的列表是按照字母顺序排列的，新建的文件中有两行内容。

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT ENTER_NAME_OF_ROOT_ELEMENT_HERE EMPTY>
```

第一行是语法要求，第二行是给的默认的例子，可以删掉，然后我们来自行输入。你发现输入<!后，就会出现几个备选的项目，其中，ELEMENT 是我们最常见的元素，第一个元素

是 XML 文件的根元素“学校”，然后要描述这个元素的数据类型，学校这个元素包含了多个学生，因此我们这样写根元素。

```
<!ELEMENT 学校 (学生+)>
```

括号中学生后面的+，说明是一个或多个学生。如果是？说明可能包含一个，也可以没有。如果是*，那么是最宽泛的定义，可以没有，可以有一个，也可以出现多次。如果什么符号都没有就是必须有一个，最多也只能有一个。

我们再来看学生，学生这个元素中会包含姓名、性别、年龄和成绩，我们将这些子元素包含到学生的声明中。

```
<!ELEMENT 学生 (姓名,性别,年龄,成绩)>
```

要注意括弧和逗号都要使用英文字符，否则 XMLSpy 在存盘的时候会报告错误。下面我们来分别声明每个子元素。

```
<!ELEMENT 姓名 (#PCDATA)>
```

#PCDATA 是数据类型，DTD 并不是强类型的语言，在 DTD 中只有两个数据类型#PCDATA 和 EMPTY，有数据和没有数据，对于 XML 的主要应用方向之一的配置文件，这样设计 DTD 无可厚非，因为配置文件对于数据类型来说没有太高的要求。

在剩下的元素中，我或许有更多的需求，性别的值只能是“男”或“女”，对不起，DTD 无法进行这么详细的约束，成绩我希望是数字，而且大于 0，小于 100，对于这个要求 DTD 也没有办法，只能写成#PCDATA。

最后有个学号的属性。

```
<!ATTLIST 学生 学号 ID #REQUIRED>
```

看写法，属性从属于一个标记，所以要列举“学生 学号”，然后是数据类型，大多数情况下有数据会在这里写成 CDATA，ID 意味着不能有重复值，还有其他并不太常用的选择，你可以到网上自行了解，其实还可以在数据类型的后面给一个默认值，我们这里不需要，#REQUIRED 是说这个值必须提供，不是必须的用#IMPLIED。

现在可以存盘，我们将它命名为 school.dtd，如果这个 DTD 有语法错误，XMLSpy 会提示，甚至会阻止你存盘，当然你可以强行的存盘。我们全部的 DTD 代码如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT 学校 (学生+)>
<!ELEMENT 学生 (姓名,性别,年龄,成绩)>
<!ELEMENT 姓名 (#PCDATA)>
<!ELEMENT 性别 (#PCDATA)>
<!ELEMENT 年龄 (#PCDATA)>
<!ELEMENT 成绩 (Java,SQL,HTML)>
<!ELEMENT Java (#PCDATA)>
<!ELEMENT SQL (#PCDATA)>
<!ELEMENT HTML (#PCDATA)>
<!ATTLIST 学生 学号 ID #REQUIRED>
```

现在你再创建一个 XML 文件，这次选择使用 DTD 文件，XMLSpy 会问你 DTD 的路径，可以使用文件选择器来选择，这样系统生成的默认内容如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE 学校 SYSTEM "C:\Users\aaa\Desktop\xml\school.dtd">
<学校>
    <学生 学号="">
        <姓名/>
        <性别/>
        <年龄/>
        <成绩>
            <Java/>
            <SQL/>
            <HTML/>
        </成绩>
    </学生>
</学校>
```

第二行<!DOCTYPE 学校 SYSTEM "C:\Users\aaa\Desktop\xml\school.dtd">将这个 XML 文件和 DTD 文件捆绑到了一起，其中 DOCTYPE 是关键，只能使用大写字母，后面是根元素标记名字，SYSTEM 指的是 DTD 在文件系统中，如果是 PUBLIC 意味着将使用互联网上的 DTD，在使用 PUBLIC 的时候，除了地址还要额外指定 DTD 的名字。

我们看到 XMLSpy 提供了符合 DTD 约束的最基本的 XML 框架，现在再次输入 school.xml 的内容，你会发现有了 DTD，XML 的录入快捷多了，因为输入<的时候，XMLSpy 会将适合的标记列出来供你选择，这当然是因为有 DTD 的作用了。

有时在一个团队中，不同任务的人都需要使用 XML，各自就会设计自己的 DTD，在项目融合的时候，有可能要将各自的 XML 合并到一起，这样难免会遇到在不同的 DTD 文件中定义的元素重名的问题，所以使用多个 DTD 的时候，可以用命名空间，在看一眼 web.xml 中根元素的标记。

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  version="3.0"  metadata-complete="true">
```

看到 web-app 后面跟着的 xmlns，这就是命名空间，xmlns 是默认的命名空间，而 xmlns:xsi 就给这个约束文件命名为 xsi，如果使用到了这个命名空间，需要写上<xsi:aaa>。

3.3.3　Schema

Schema 是 DTD 的替代品，它的作用也是用来约束 XML 文件的，那么既然有了 DTD，为什么还要 Schema 呢？我想你也能感觉到 DTD 的弱点，你先自己总结一下 DTD 的弱点是什么？我能想到的有这样几个方面，数据类型不丰富，DTD 本身的语法规则和 XML 不同，其实还有其他

的不同，但是你能理解到那，我就涉及到那。

因为 DTD 存在着不足，才出现了 Schema，所以 Schema 一定解决了这些问题，计算机软件领域到处都充斥着这个逻辑。

我们现在就根据 school.xml 的内容来创建 Schema，通过这个过程体会 Schema。Schema 文件的扩展名是 xsd，我们看到有两个 xsd 文件可供选择，因为现在存在着两个版本的 Schema 标准，W3C 只接受了部分的标准，建议使用 W3C 的标准，这样会有更好的通用性。

与 DTD 相比 Schema 编写起来要复杂多了，所以创建 Schema 时，XMLSpy 自动提供了方便的图形编辑界面，我们要通过编写代码来体会 Schema，所以需要选择“文字”模式。如图 3-16 所示。

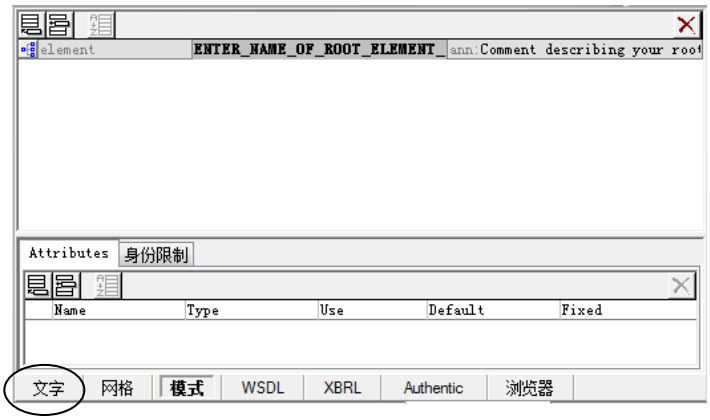


图 3-16

我们先来分析根元素的定义，Schema 有丰富的表达方式，完成同一个任务可能有不同的写法，在这里我选择相对清晰，适合手工编写的写法来讲解，一旦理解了这个写法，其他的写法也很容易理解。具体代码如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
                                    elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:element name="学校" type="学校类型"/>

    <xs:complexType name="学校类型">
        <xs:sequence>
            <xs:element  name="学生"  type="学生类型"  maxOccurs="unbou
nded"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

首先我们看到，Schema 完全符合 XML 语法规则，有第一行的声明，也有单根结构，从<xs:element name="学校">这行开始，我们来声明 school 的根元素“学校”，数据类型是自定义的

“学校类型”。

下面来定义学校类型，<xs:complexType >说明“学校类型”是个复杂类型，在 Schema 中有简单类型和复杂类型的区分，复杂类型就是包含了子元素的元素，简单类型是仅包含文本的元素。

<xs:sequence>的意思是以下元素按顺序出现，如果是<xs:choice/>那么就是选择出现，到目前按顺序出现也没什么实际意义，因为只可能出现子元素“学生”。

<xs:element name="学生" type="学生类型" maxOccurs="unbounded"/>，这里描述子元素“学生”，数据类型是“学生类型”，最多允许出现无穷多，这时开始表示出 Schema 的强大了，我们可以这样设置<xs:element name="学生" type="学生类型" **minOccurs="10" maxOccurs="50"**/>，这是定义学校中最少要有 10 个学生，最多是 50 个学生，相比 DTD 的*+?，至少在这一方面上强大多了。由于使用了 XMLSpy，所以有哪些能够定义的功能是显而易见的，因为输入空格后，提示就会出现。

下面我们要定义“学生类型”，具体代码如下。

```
    <xs:complexType name="学生类型">
        <xs:sequence>
            <xs:element name="姓名" type="姓名类型"/>
            <xs:element name="性别" type="性别类型"/>
            <xs:element name="年龄" type="年龄类型"/>
            <xs:element name="成绩" type="成绩类型"/>
        </xs:sequence>
        <xs:attribute name="学号" type="xs:integer" use="required"/>
 </xs:complexType>
```

大部分内容你应该能够理解，在后面添加了对于学号属性的描述，作为属性总是会声明成简单类型，如果没有更多的约束，可以直接使用，我这里使用了 xs:integer，你可以想象还会有 xs:string 之类的，事实上原本的 Schema 几乎支持数据库能够支持的所有数据类型，但是，W3C 的标准只采纳了其中的一个部分。如果有进一步的约束，可以定义类型为“学号类型”，在后面再对“学号类型”进行进一步的描述，但是“学号类型”也得是简单类型。use="required"的含义等同于 DTD 中属性必须提供的定义#REQUIRED。

下面来定义“姓名类型”。具体代码如下。

```
 <xs:simpleType name="姓名类型">
        <xs:restriction base="xs:string">
            <xs:minLength value="2"/>
            <xs:maxLength value="5"/>
        </xs:restriction>
 </xs:simpleType>
```

终于定义到简单类型了，上面代码的第二行说明了这个类型的基础是字符串，然后对字符串的最小长度和最大长度进行了约束，现在越来越能看出 Schema 的强大了。

这个 Schema 剩下的代码如下。

```
 <xs:simpleType name="性别类型">
```

```
    <xs:restriction base="xs:string">
        <xs:enumeration value="男"/>
        <xs:enumeration value="女"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="年龄类型">
    <xs:restriction base="xs:integer">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="100"/>
    </xs:restriction>
</xs:simpleType>

<xs:complexType name="成绩类型">
    <xs:choice>
        <xs:element name="Java" type="分数类型"/>
        <xs:element name="SQL" type="分数类型"/>
        <xs:element name="HTML" type="分数类型"/>
    </xs:choice>
</xs:complexType>

<xs:simpleType name="分数类型">
    <xs:restriction base="xs:integer">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="100"/>
    </xs:restriction>
</xs:simpleType>
```

在性别类型中，我枚举了男和女。在年龄类型中我限定了范围从 0 到 100 岁，限定范围支持小于和小于等于。成绩类型是复杂类型，考虑到有人有可能没有三门全考，所以使用了<xs:choice>，三门课程的数据类型都是“分数类型”，而“分数类型”是简单类型。

前面讲到 Schema 会有多种写法，我只不过解释了一种写法，现在列举另外的一种写法，你可以自行分析一下，未来万一要读别人写的 Schema 文件，不至于太陌生。

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"xmlns:altova="http:
//www.altova.com/xml-schema-extensions"
    elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:element name="学校">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="学生" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="姓名">
                                <xs:simpleType>
                                    <xs:restrict base="xs:string">
                                        <xs:min Length value="2"/>
```

```
                        <xs:max Length value="5"/>
                    </xs:restric tion>
                </xs:simpleType>
            </xs:element>
            <xs:element name="性别">
                <xs:simpleType>
                    <xs:restrict ion base="xs:string">
                        <xs:enu meration value="男"/>
                        <xs:enu meration value="女"/>
                    </xs:restrict ion>
                </xs:simpleType>
            </xs:element>
            <xs:element name="年龄">
                <xs:simpleType>
                    <xs:restricti on base="xs:integer">
                        <xs:min Inclusive value="0"/>
                        <xs:max Inclusive value="100"/>
                    </xs:restrict ion>
                </xs:simpleType>
            </xs:element>
            <xs:element name="成绩">
                <xs:complexType>
                    <xs:choice>
                    <xs:element name="Java" type="分数类型"/>
                    <xs:element name="SQL" type="分数类型"/>
                    <xs:element name="HTML" type="分数类型"/>
                    </xs:choice>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="学号" type="xs:integer"/>
    </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>

<xs:simpleType name="分数类型">
    <xs:restriction base="xs:integer">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="100"/>
    </xs:restriction>
</xs:simpleType>
</xs:schema>
```

其实两种写法的区别无非是将类型的定义单独写，还是直接写在元素使用的里面。

通过这个例子，你应该对 Schema 有了基本了解了，我并没有全面的介绍 Schema，因为这是一个不小的领域，无论是源于我的主观还是局限，很多内容我都没涉及到，事实上我只希望能够

通过这个过程带领你了解 Schema，成为 Schema 的高手，你还需要了解这方面更多的知识。

我们已经体验了 DTD 和 Schema 的差别，无疑 Schema 要强大得多，现在也越来越趋向于使用 Schema 了，但是由于 Schema 编写起来比 DTD 复杂的多，所以 DTD 一直难以淘汰，我们现在很多应用 XML 做配置文件的场合还在广泛的使用 DTD，因为应对配置文件，DTD 够了。

但是因为 XML 有时会被作为轻量级的数据库，这样对数据类型和数据类型约束的需求，都只有 Schema 能够胜任，不但这样，新的语言技术常常将数据库中的数据以 XML 的形式组织在内存中，这时都会看到 Schema 的应用。

3.3.4 CSS 和 XSL

XML 是个典型的，存放数据的文件，但是现在只要见到 XML，人们都会联想到 HTML，毕竟样子太像了，HTML 的目的不是存放，而是在浏览器中展现出来，随着 CSS 越来越广泛的应用，人们发现 HTML+CSS 的结果是，HTML 越来越像 XML，被用来存放数据了，如何显示，取决于 CSS 文件，何不干脆用 XML 代替 HTML，将 HTML+CSS 的模式转变为 XML+CSS，这样 XML 就负责存放数据，CSS 负责告诉不同的显示终端该如何显示。

由于大量的 HTML 程序员喜欢随性的编写风格，所以放弃 HTML，改用 XML 对于这些人来说是痛苦的。但是未来的趋势，人们更倾向于严谨的编写符合 XML 规范的网页文件，达到这一要求的网页被称为 XHTML。事实上 XML+CSS 完全能够替代 HTML，但是由于互联网上存在着太多的松散的 HTML 网页，所以提出了过渡的技术 XHTML，最终的目的是使用 XML 来制作网页。

因为前面学习 HTML 的时候已经大量练习了 CSS，所以在这里我不再讨论具体的 CSS 技术。

如同 Schema 是 DTD 的替代品一样，XSL 是 CSS 的替代品，看起来 CSS 也有一些值得改进的地方，事实上 CSS 有严重的局限性，CSS 通常用来控制输出的大小、颜色和位置之类的与纯显示相关的属性。

如果我们要对显示的内容进行判断、排序和筛选，那么就要用到 XSL 技术。从另一个角度看，CSS 针对的是标记，而 XSL 不仅仅可以针对标记，还可以针对内容进行判断。

在我的讨论中或许存在着一个暗示，CSS 和 XSL 技术似乎是将 XML 转换成 HTML，以便将内容显示到浏览器上的技术，不可否认这是现在主要的功能，但是并不局限于这个方面，XSL 也用来将 XML 内容转换成其他的表现形式。

XSL 事实上是由几个技术结合在一起工作的，但是我们还是抛弃掉那些学术的分类，只要你最后知道可以用什么就好了。

现在我们尝试着用 XSL 将 school.xml 显示成网页。先将 XSL 文件写成下面这个样子，以便学习如何将 XSL 和需要转换的 XML 联系到一起。具体代码如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet  version="1.0"  xmlns:xsl="http://www.w3.org/1999/XSL/
Transform" xmlns:fo="http://www.w3.org/1999/XSL/Format">
```

```
<xsl:template match="/">
    <html>
        <body>
            <h1>学生信息</h1>
            <table border="1">
                <tr>
                    <th>学号</th>
                    <th>姓名</th>
                    <th>性别</th>
                    <th>年龄</th>
                    <th>Java</th>
                    <th>SQL</th>
                    <th>HTML</th>
                </tr>
            </table>
        </body>
    </html>
</xsl:template>
</xsl:stylesheet>
```

首先我们判断，这是一个非常纯粹的 XML 文件，xsl:stylesheet 是根元素，这个 XML 使用命名空间引入了约束文件，来看我加粗的标记<xsl:template match="/">，这个标记是必须的，用于将 XML 文件读到内存中，所有的其他操作都是在这个基础上进行的，match 属性的/指的是从根元素开始读。其余的部分没有什么特别的，都是标准的 HTML 标记，事实上这个 XSL 文件还什么都没干，只是写了一个框架。

我们将下面的一句话加入到 school.xml 的第二行，以便让刚刚写好的 XSL 文件起作用。

```
<?xml:stylesheet type="text/xsl" href="school.xsl"?>
```

现在用浏览器访问 school.xml，页面显示如图 3-17 所示。注意，访问的是 xml 文件，XMLSpy 本身带了浏览器，你可以任意选择使用那个浏览器。

图 3-17

我们发现显示的就是在 XSL 中写的那些 HTML 内容，XML 文件中的内容完全没有显示出来，我们可以得出一个结论，显示什么是由 XSL 文件决定的，如果 CSS 什么都不做，那么所有的东西都会显示，顶多是没有什么漂亮的效果，而 XSL 如果什么都不做，那么就什么都不会显示了。具体代码如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet  version="1.0"  xmlns:xsl="http://www.w3.org/1999/XSL/
Transform" xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <xsl:template match="/">
```

```
        <html>
            <body>
            <xsl:value-of select="学校/学生/@学号"/>
            <xsl:value-of select="学校/学生/姓名"/>
            </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

上面的代码，显示了一个 001，是第一个学生的属性学号的值，属性前面加上@进行识别。第二句显示的是张三，第一个学生的姓名，是取得 XML 中元素值的办法，因为是从根元素装载 XML 文件，所以要从根元素开始写出到达某一节点的路径。

现在我们要学习如何针对学生进行循环。具体代码如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet  version="1.0"  xmlns:xsl="http://www.w3.org/1999/XSL/
Transform" xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <xsl:template match="/">
        <html>
            <body>
                <xsl:for-each select="学校/学生">
                    <xsl:value-of select="@学号"/>
                    <xsl:value-of select="姓名"/>
                </xsl:for-each>
            </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

这样就能将所有的学号和姓名都显示出来了，for-each 一看就知道是循环，后面的路径是循环哪一层，由于有了 for-each，我们再取值就会省略循环体已经描述清楚的路径。现在你尝试着按照前面我描写的表头，将整个 XML 文件中的值填入到表格中，具体代码如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet  version="1.0"  xmlns:xsl="http://www.w3.org/1999/XSL/
Transform" xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <xsl:template match="/">
        <html>
            <body>
                <h1>学生信息</h1>
                <table border="1">
                    <tr>
                        <th>学号</th>
                        <th>姓名</th>
                        <th>性别</th>
                        <th>年龄</th>
                        <th>Java</th>
                        <th>SQL</th>
                        <th>HTML</th>
```

```
                </tr>
                <xsl:for-each select="学校/学生">
                    <tr>
                        <td><xsl:value-of select="@学号"/></td>
                        <td><xsl:value-of select="姓名"/></td>
                        <td><xsl:value-of select="性别"/></td>
                        <td><xsl:value-of select="年龄"/></td>
                        <td><xsl:value-of select="成绩/Java"/></td>
                        <td><xsl:value-of select="成绩/SQL"/></td>
                        <td><xsl:value-of select="成绩/HTML"/></td>
                    </tr>
                </xsl:for-each>
            </table>
        </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

代码效果如图 3-18 所示。

学生信息

学号	姓名	性别	年龄	Java	SQL	HTML
001	张三	男	20	85	90	79
002	李四	男	19	87	76	90
003	王五	男	21	73	82	45

图 3-18

现在进一步的要求是将不及格的成绩显示成红色，我们要用到判断语句。以下是判断成绩是否及格的部分代码。

```
<xsl:if test="成绩/Java &lt;= 60">
    <td style="color:red;"><xsl:value-of select="成绩/Java"/></td>
</xsl:if>
<xsl:if test="成绩/Java &gt; 60">
    <td><xsl:value-of select="成绩/Java"/></td>
</xsl:if>
<xsl:if test="成绩/SQL &lt;= 60">
    <td style="color:red;"><xsl:value-of select="成绩/SQL"/></td>
</xsl:if>
<xsl:if test="成绩/SQL &gt; 60">
    <td><xsl:value-of select="成绩/SQL"/></td>
</xsl:if>
<xsl:if test="成绩/HTML &lt;= 60">
    <td style="color:red;"><xsl:value-of select="成绩/HTML"/></td>
</xsl:if>
<xsl:if test="成绩/HTML &gt; 60">
```

```
    <td><xsl:value-of select="成绩/HTML"/></td>
</xsl:if>
```

3.3.5 DOM

我们现在学习过的内容包括 XML 数据文件、DTD 和 Schema 这两个约束文件、CSS 和 XSL 转换文件，还有一组需求。用程序读写 XML，要说这不该是个问题，因为大多数编程语言都有 IO 流支持，但是深入的看就会发现，写 XML 用 IO 流相对容易，可是读却不那么容易，因为我们的需求往往不仅仅要有读取文件内容的能力，大多数情况下，我们需要取得某一特定标记的值，如果单纯依赖 IO 流，每次就需要写比较复杂的分析程序，而这些代码完全可以重用。

因此 W3C 提出了一个统一的标准 DOM，DOM 的思想是将 XML 文件读到内存中，使其形成一个对象，这个对象内在映射着 XML 文件的树形结构，我们可以通过调用对象的成员方法来访问这些内容。

W3C 做这件事情还有一个巨大的好处，DOM 不是建立在某个编程语言基础上的，作为一个国际标准，几乎所有的编程语言都实现了这个标准，这样不同语言利用 DOM 访问 XML 的做法就基本一致了，当然完全的一致并不现实，毕竟语言不同，但是至少如果你能使用 Java 实现 DOM，那么改成 C#来做同样的事情，只需要看一眼 C#的例子就可以了。

现在我们来看基于 Java 的 DOM 如何解析 XML，读取文件大家都知道要使用 java.io.*，这是最底层的 IO 操作手段，作为更高级方便的 DOM，自然有更高级的类来包装 IO 流了，它来自 javax.xml.parsers.*，但是我们还要导入 org.w3c.dom.*，为什么设计者要将 DOM 的操作类放到两个地方呢？我们看第二个导入，很明显 w3c 说明这是对 W3C DOM 的实现，但是这个标准不关心具体的实现语言，那么语言要做些初期工作，比如要将 XML 文件读到内存中，甚至要形成可供操作的对象，所以有些类被放到第一个导入中了。具体代码如下。

```
package com.wy.dom;

import javax.xml.parsers.*;
import org.w3c.dom.*;

public class MyDOM {
    public static void main(String[] args) {
        try{

        }catch(Exception e){}
    }
}
```

这样说来，我们的语句也将分成两个部分，第一个部分是读取 XML 文件形成对象，第二个部分是操作对象。现在我们先来看第一步。具体代码如下。

```
package com.wy.dom;

import javax.xml.parsers.*;
```

```
import org.w3c.dom.*;

public class MyDOM {
    public static void main(String[] args) {
        try{
    DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
    DocumentBuilder db = dbf.newDocumentBuilder();
    Document doc = db.parse("school.xml");
        }catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

比想象的要复杂多了。一行行来看，很明显这里使用了工厂模式，再回顾一下什么是工厂模式，为什么要用工厂模式，我们创建对象的过程太复杂了，就会想到使用工厂模式。到了第三步，我们得到了一个 Document 的对象，Document 是 org.w3c.dom 中的类，这个时候我们就完成了属于 Java 特有的操作。

school.xml 没有给路径，这就要求文件和 Java 程序在一个目录下，如果在正确的位置找不到这个文件，就会在控制台打印异常。你自己尝试一下需要将这个 xml 文件放到什么地方，实在不行的话，使用绝对路径吧。

现在我们来研究 Document 的操作，你自己用 doc 点点，然后研究一下里面的成员方法，能够看出这个类的成员方法相当的多，这是个强大的类。在上面代码获得了 Document 对象的后面加入下面的话。

```
NodeList root = doc.getElementsByTagName("*") ;
```

这句话的作用是，从根元素开始，将整个 doc 解析到 NodeList 对象中。NodeList 顾名思义是节点的列表，或者说是节点的集合，不知道你现在头脑中有没有那个树形结构。

NodeList 中，成员方法很少，有意义的就是 item，这个方法返回值是 Node，需要一个数字的参数，估计是第几个节点，我们取第 0 个节点试试。

```
Node node = root.item(0) ;
System.out.println(node.getNodeName()) ;
```

没错打印出来是“学校”，即我们根节点的名字，修改数字为 1、2、3 试试。结果和我们想象的一样吧，就是 XML 中每个标记的名字。

但是这样的设计有一个问题，如果我要找到每个学生的姓名是很麻烦的，原因是原本的树形结构变成了线性结果，这样第一个姓名是下标 2，第二个是 10，他们之间的间隙和中间的元素数量有关，可是在很多时候两个元素之间的间隔不确定，比如，如果第一个学生的成绩中只有两项，那么第二个姓名的下标就成了 9。

如何在学生这一层进行循环是现在要解决的问题。我们发现 Node 有丰富的成员方法，其中有个方法是 getChildNodes，返回值又是一个 NodeList，从方法名来看这个 NodeList 是下一层子

节点，如果“学校”的下一层子节点就是“学生”，我们来看看是不是这样，当然要用 item，输入 item(0)，然后看节点的名字，很让人失望，别灰心换成 item(1)试试看。

```
NodeList root = doc.getElementsByTagName("*");
Node node = root.item(0);
NodeList n1 = node.getChildNodes();
System.out.println(n1.item(1).getNodeName());
```

没错是学生，那么 2、3、4、5 的结果如何呢？我们看到了希望的“学生”，奇怪的是“学生”的下标跳一个才有效，奇数的下标才能得到“学生”，偶数的结果是“#text”，其实这个位置是节点之间的间隙，我们可以用奇数循环，也可以在取得 NodeName 后用 equals 进行筛选。

还不能完成代码，因为虽然我们能够取得节点的名字，但是还不了解如何得到其中的值，“学生”这一层是没有值的，要到“姓名”这一层取值。

```
NodeList root = doc.getElementsByTagName("*");
Node node = root.item(0);
NodeList n1 = node.getChildNodes();
System.out.println(n1.item(1).getChildNodes().item(1).getNodeName());
```

这样我们得到的是“姓名”，将 getNodeName 换成 getNodeValue 试试，得到的是 null，不是我们期望的值，这是因为 DOM 认为值是下一层的内容，所以这句话要变成如下代码。

```
n1.item(1).getChildNodes().item(1).getChildNodes().item(0).getNodeValue()
```

你发现我这次用的是下标 0，还记得前面用 0 的时候，NodeName 的结果是#text，这个名字包含的就是内容，就是元素的值。考虑到取值的操作相当的普遍，所以 Node 还提供了 getFirstChild 方法，效果和 getChildNodes().item(0)一样，这样这句话就变成如下代码。

```
n1.item(1).getChildNodes().item(1).getFirstChild().getNodeValue()
```

现在我们知道如何循环，如何取值了，那么请将所有的姓名都打印出来吧。具体代码如下所示。

```
package com.wy.dom;

import javax.xml.parsers.*;
import org.w3c.dom.*;

public class MyDOM {
    public static void main(String[] args) {
        try{
            DocumentBuilderFactory dbf =
                DocumentBuilderFactory.newInstance();
            DocumentBuilder db = dbf.newDocumentBuilder();
            Document doc = db.parse("school.xml");

            NodeList root = doc.getElementsByTagName("*");
            Node node = root.item(0);
            NodeList n1 = node.getChildNodes();
            for(int i = 1; i < n1.getLength(); i +=2){
```

```
                System.out.println(n1.item(i).getChildNodes().
                item(1).getFirstChild().getNodeValue());
            }
        }catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

请注意那个循环语句，我们产生的是奇数的循环，这是因为我知道奇数才有效，事实上有比这个更常用的做法。具体代码如下。

```
package com.wy.dom;

import javax.xml.parsers.*;
import org.w3c.dom.*;

public class MyDOM {
    public static void main(String[] args) {
        try{
            DocumentBuilderFactory dbf =
                DocumentBuilderFactory.newInstance();
            DocumentBuilder db = dbf.newDocumentBuilder();
            Document doc = db.parse("school.xml");

            NodeList root = doc.getElementsByTagName("*");
            for(int i = 0 ; i < root.getLength() ; i ++){
                if(root.item(i).getNodeName().equals("姓名")){
                    System.out.println(root.item(i).
                            getFirstChild().getNodeValue());
                }
            }
        }catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

一来这样做相对简洁，二来大多数的需求是得到指定标记的值，所以第二种做法更常用。

现在我们再来看看用 JavaScript 如何使用 DOM 来解析 XML。在我提供 JavaScript 代码前，我想请你看看这句话，doc.getElementsByTagName("*")，这是我取的 Java 代码，是不是很熟悉，在 JavaScript 操作 CSS 的时候，我们用过 Document.getElementById，只不过后来我们都用了简略的写法，事实上 JavaScript 中，Document 对象也有 getElementByTagName 方法，这是因为其实我们早就在试用 JavaScript 的 DOM 了，因为对于 JavaScript 来说，HTML 本身就是 XML，只不过不那么规范。

现在我来提供如下代码，像 Java 那样解析 school.xml。

```
<html>
    <body>
        <script>
        if(window.ActiveXObject){// IE 浏览器支持
            obj = new ActiveXObject("Microsoft.XMLDOM");
            obj.load("school.xml");
        }
        alert(obj);
        </script>
    </body>
</html>
```

因为现在流行的浏览器有微软的 IE，还有 Mozillia 和 FireFox，它们内建的对 DOM 的支持对象不同，所以先要判断是那种类型的浏览器，针对不同的浏览器要用不同的文件加载方式。这里我以 IE 浏览器的代码为例子，在 if 判断语句结束后，我们用 alert 显示 object 是否是对象，经过以上操作，我们将 XML 读到内存中，并且形成了 DOM 对象 object。具体代码如下。

```
<html>
    <body>
        <script>
        if(window.ActiveXObject){
            obj = new ActiveXObject("Microsoft.XMLDOM");
            obj.load("school.xml");
            root = obj.getElementsByTagName("*");
            alert(root[0].childNodes[0].childNodes[0].
                childNodes[0].nodeValue);
        }
        </script>
    </body>
</html>
```

通过 getElementsByTagName("*")从根元素开始获得 NodeList，alert 语句显示的是第一个姓名，root[0]是学校这一层节点，childNodes[0]是学生这一层，第二个 childNodes[0]到达姓名这一层，第三个 childNodes[0]，类似于 Java 中的 getFirstChild()方法，然后是取值的操作。

在 JavaScript 中.nodeValue 被称为属性，因为所处的位置，我们要从这个属性中取值，事实上内部调用了 getNodeValue 方法，如果要给这个属性赋值，那么内部会通过调用 setNodeValue 来实现，Java 也有类似的概念，但是没有其他更新的语言使用的那么彻底。

相比 Java，JavaScript 在节点树上遍历也显得简洁多了，因为 item 的操作被类似于数组的表达代替了。

使用 JavaScript 来实现 DOM 通常有多种方法，这里不一一陈述。有时对 HTML 网页中的内容也会用到经过变化的 DOM，比如操作表格，这个在后面具体的任务中，我会涉及。

3.3.6　SAX

XML 技术的有趣之处就是总是有两个相同功能的不同技术，比如 DTD 和 Schema、CSS 和 XSL，现在是 DOM 和 SAX，DOM 和 SAX 的功能一样，都是用来解析 XML 文件的，不同之处是 DOM 一次性将整个 XML 文件读到内存中，并且包装成对象，而 SAX 每次只读取一行，处理完后将抛弃掉这一行，然后处理下一行，这样做的好处是不占用太多内存，如果 XML 作为数据库应用，那么有可能内容很多，文件很大，使用 DOM 或许会超出内存的大小，这时便只能使用 SAX，但是这样也说明 SAX 应用的不如 DOM 频繁，因为读取很大的 XML 文件的机会并不多。SAX 的另一个弱点是，因为数据会被丢弃，所以和 DOM 相比没法再次使用数据。

SAX 使用了类似于事件机制来处理数据，一行数据读进来会引发一个事件，事件处理程序负责处理这行数据，我们来看如下代码。

```
import org.xml.sax.*;
import org.xml.sax.helpers.DefaultHandler;
import javax.xml.parsers.*;
import java.io.*;

public class wyXMLSAX extends DefaultHandler {
boolean b = false;
    //该构造方法读取 XML 文件，并加入事件机制
    public wyXMLSAX() {
        try {
            SAXParserFactory spf = SAXParserFactory.newInstance();
            SAXParser sp = spf.newSAXParser();
            sp.parse(new File("school.xml") ,this);

        }
        catch (Exception ex) {
            ex.printStackTrace();
        }
    }
    public static void main(String args[]){
        wyXMLSAX wxs = new wyXMLSAX();
    }
//每读入一个元素，就会调用一次该方法
    public void startElement(String url , String localName,
            String qName , Attributes attributes)  {
        if(qName.equals("姓名")){
            b = true;
        }else {
            b = false;
        }
    }
    public void characters(char[] ch , int start , int length){
```

```
            try {
                String ss=new String(ch , start , length);
                if(b){
                    System.out.println (ss);
                }
            }
            catch (Exception ex) {
            }
        }
    }
```

我本可以更加详细地讲解上面的代码是如何工作的，但是这个相对使用较少的技术点，恰好提供了一个由你自己探索的机会，你可以将 startElement 方法里面的那些 String 参数打印出来看看，自己分析它们是什么，要注意在不同的环境下，有些参数可能无效。你最好能够写下一些东西，将我的这个部分完善起来。

3.3.7 XML 总结

XML 文件负责存放数据，DTD 和 Schema 负责约束，CSS 和 XSL 负责转换成表现形式，DOM 和 SAX 负责解析，这些技术形成了完整的 XML 体系，软件开发领域在可能的情况下逐渐皈依了 XML。

我们来看 Tomcat 是如何使用 XML 的，在 Tomcat 的运行过程中会多次使用 XML，我们先忽略掉我没有介绍到的内容，就看如何通过 web.xml 来启动 Servlet，我们回顾一下，在 web.xml 中，我们写入了什么？一个 servlet 元素和一个 servlet-mapping 元素，在 servlet 中有 servlet-name 和 servlet-class 两个元素，Tomcat 会使用 DOM 读取这个 web.xml 文件，找到其中所有的 servlet 元素，读取其中的值，然后利用反射机制创建 class 的对象。

这个 DOM 的对象将一直存放在 Tomcat 的内存中，当浏览器发出了请求，Tomcat 会判断是静态的 HTML 文件，还是 Servlet，如果是 Servlet，它就会检索所有的 servlet-mapping 元素，通过 url 的匹配找到 servlet-name，进而调用那个对象的 service 方法。

那么如何确保 Servlet 的开发人员一定能够正确配置 web.xml 呢？我们能看到这个文件使用的命名空间引入了 DTD 或 Schema，很多初级的程序员在遇到一个 XML 文件时，会期望得到这个 XML 使用的说明，其实资深的程序员会寻找对应的 DTD 或 Schema，有了约束文件自然知道有什么标记，能够使用什么标记。

学习 XML 的目的是为了更好地理解 Tomcat 的工作而扫除障碍，我不希望你成为一个只知道该如何做，而不知道为什么要这样做的程序员，或许有一天你要编写类似于 Tomcat 的软件，至少有可能成为项目经理，去管理和配置一个软件项目的架构，那么今天来看你最基本的一个能力就是能够驾驭 XML。

3.4　购物网站的商品展示

类似于购物网站的商品展示页面十分常见，我以京东商城的商品为例子，看一眼京东的网站就能意识到这将是一个对于我们来说非常庞大的工程，我们绕开首页，来到一个具体类别商品的页面，我选取的是笔记本电脑的展示页面。你有选择你喜欢的商品的自由，甚至去模拟实现其他购物网站的相关页面。

具体到某个类别的页面对于我们来说太复杂了，虽然我们最终能够实现所有的一切，但是现在我还是尽可能的简化任务，以便让你能更容易上手，所以现在我选取了完全的商品展示部分。如图 3-19 所示。

好吧，这就是我要做的东西，具体的页面你有必要到京东的网站上自己体验一下。看完目标，我们静下来思考一下都有那些事情要做。看起来我们需要设计数据库来存放这些商品的信息，需要读取数据并进行页面设计，由于商品比较多，我们需要分页显示。再次说明，我现在本着尽可能简化的原则，否则还需要后台的商品录入程序。

图 3-19

3.4.1　数据库设计

首先我们要收集什么数据需要存放在数据库中。数据库设计这个工作非常重要，因为未来将有很多程序和现在所设计的数据库发生关系，虽然这个设计将来不可避免的会有所调节，但是我们希望能够尽可能的一劳永逸，因此将什么数据放到数据库表中时，我们应该超越现在要完成的工作，尽量的收集全数据。但是让人纠结的是，为了减轻学习过程中重复性工作的负担，我还希望尽可能的简化工作量，要知道设计完数据库，我们还要录入一定数量的基础数据，以便能够展示出来效果。权衡之后，我选择了简化，但是我希望你能够比我多想一些。

在这个页面上能够看到有商品图片、商品名称、价格、赠品和直降标记。每个商品下面都有评论数，这个应该是计算得来的，暂且说明成评论。鼠标放到图片或是商品名称处，会出现一个提示，提示的内容是促销信息。从数据库的角度看，我们还需要商品编号，事实上在购买按钮中隐藏着对购买编号的使用，我们现在就先存放这些。

事实上要实现京东商城的网站，在商品相关的表中要存放的数据类别是非常庞大的，还会有商品品牌、库存地点、销量，大量的性能参数、商品评分、优惠额度、赠品、包装清单和售后服务等。

现在我们先来看简化的结果，如图 3-20 所示。

编号	名称	价格	赠品	直降	评论	图片	促销

图 3-20

从三范式的角度看，评论和编号不是主键依赖关系，需要分离出去，这样我们就有了两个表，一个是商品表：create table products (pid int primary key , name varchar(100) not null , price double(8,2) , gift int , down char(1) , img varchar(20) , info varchar(100)) ;。一个是评论表：create table comment (pid int references products.pid , mess varchar(100)) ;

其中赠品 gift 我设计成了 int，这个对应 pid，在需要的情况下可以直接指向赠品的 pid，在我们这个程序中，暂时不需要这样的操作，我暂时设置如果没有赠品这里就存放 0，如果有就存放 100000，直降 down 是 char(1)类型的，如果有直降，该列的值是'y'，否则是'n'。

通常的做法图片会以文件的形式存放，数据库中存放文件的路径，这样的好处是在数据库管理的时候相对方便，备份数据的时候效率更高，而且相同的图片只用存一个，比较节约存储空间。

现在我们要录入一些数据，这些数据来源于我们模拟的网站，为了实现分页的效果，至少录入 25 个商品，这是个不小的工作量，我在光盘中提供了数据录入的代码，你直接将其复制到 MySQL 的客户工具中运行即可录入 25 个商品。但是如果你要实现其他类型的商品，这个工作你要自己完成。

3.4.2 展示页面程序

我们先写出这个 Servlet 程序的基础，并且配置好 web.xml 文件。首先我在 Tomcat 的 webapps 目录中创建一个新的目录，我给它命名为 365buy，在这个目录中创建子目录 WEB-INF，并且找到一个 web.xml 复制到这个目录中，然后在 WEB-INF 中创建子目录 classes，编译好的 Java 类文件就放到这个子目录中。

在编译器中，我给这个 Servlet 类命名为 Products，在这一步我设置好中文输出，准备输出流，不能每次都到 response 里获取输出流，然后连接上数据库，并且输出基本的 HTML 标记。具体代码如下。

```
package com.wy;
```

```
import javax.servlet.http.*;
import java.io.*;
import java.sql.*;

public class Products extends HttpServlet {
    public void doGet(HttpServletRequest request,
HttpServletResponse response) {
        try {
            // 设置中文显示
            response.setContentType("text/html;charset=gb2312");

            // 准备输出流
            PrintWriter out = response.getWriter();

            // 准备数据库连接
            Class.forName("org.gjt.mm.mysql.Driver");
            Connection cn = DriverManager.getConnection(
            "jdbc:mysql://127.0.0.1:3306/360buy", "root", "123456");

            // 输出 HTML 基本标记
            out.println("<html>");
            out.println("<body>");
            out.println("</body>");
            out.println("</html>");
        } catch (Exception e) {}
    }
}
```

然后在 web.xml 中添加对这个 Servlet 的配置。具体代码如下。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  version="3.0"
  metadata-complete="true">

  <display-name>Welcome to Tomcat</display-name>
  <description>
     Welcome to Tomcat
  </description>

  <servlet>
   <servlet-name>show</servlet-name>
   <servlet-class>com.wy.Products</servlet-class>
  </servlet>
```

```
  <servlet-mapping>
   <servlet-name>show</servlet-name>
   <url-pattern>/show.bin</url-pattern>
  </servlet-mapping>

</web-app>
```

再来重复一下部署的过程，我们找到编译出来的 class 文件，连同生成的包目录复制到 classes 中，如果 Tomcat 在运行，就将它关掉，然后启动 Tomcat，这时可以到浏览器上访问 http://127.0.0.1:8080/365buy/show.bin，你可能什么都看不见，在浏览器上单击鼠标右键，在右键菜单中会有“查看源文件”，选择它将看到一个 HTML 文件，这个文件中有你生成的 HTML 语句。这是个重要方法，现在我们编写的语句不是直截了当的 HTML 语句，而是通过 Java 语言生成的 HTML，在这个过程中有可能是疏忽，有可能就是晕了，生成的 HTML 很有可能和我们的想象不同，当网页效果不是我们想要的，那么看一下源文件会直接一点。

问题是如果源文件确实有问题，那么可能的原因一个是我们 Java 的代码不对，还有一个新手常见的错误，就是部署过程的问题，我们有没有将新版本复制到正确的位置，有没有重启 Tomcat，当你不知道问题发生在什么地方的时候，最好的做法不是回想那里出了问题，而是重新部署一遍。

假设没有问题了，我们来看如何安放这些商品，我们参照的网页是每行 4 个商品，我们也这样做，有不同的办法来做这样的布局，我用表格，表格一定不是我一点点写出来的，而是有一个商品就要一个单元格，这样我要用程序产生 td 标记，而且每出现 4 个 td，就加入一个 tr。

这样看来在这一步我有两个工作要做，一个是查询到数据，一个是通过循环创建表格。为了清楚地看到具体效果，我将表格线显示出来。具体代码如下。

```
package com.wy;

import javax.servlet.http.*;
import java.io.*;
import java.sql.*;

public class Products extends HttpServlet {
    public void doGet(HttpServletRequest request,
            HttpServletResponse response) {
        try {
            // 设置中文显示
            response.setContentType("text/html;charset=gb2312");

            // 准备输出流
            PrintWriter out = response.getWriter();

            // 准备数据库连接
            Class.forName("org.gjt.mm.mysql.Driver");
            Connection cn = DriverManager.getConnection(
                "jdbc:mysql://127.0.0.1:3306/360buy", "root", "123456");
```

```
            // 查询数据
            Statement st = cn.createStatement();
            ResultSet rs = st.executeQuery(
"select img , name , price , gift , down , info from products");

            // 输出 HTML 基本标记
            out.println("<html>");
            out.println("<body>");
            out.println("<table width='100%' border='1'>");

            int count = 0;
            while (rs.next()) {
                if (count % 4 == 0) {// 每四个商品一行
                    out.println("<tr>");
                }
                out.println("<td align='center'>");

                // 请理解标记结束的逻辑，为什么要先++再判断
                count++;
                if (count % 4 == 0) {
                    out.println("</tr>");
                }
            }
            out.println("</tr>");
            out.println("</table>");
            out.println("</body>");
            out.println("</html>");
        } catch (Exception e) {}
    }
}
```

还要重复部署的过程，这个过程是不是很烦，那么你想想有没有简化这个过程的办法。

你看到为了避免和 Java 语言字符串定义时使用的双引号冲突，HTML 的属性我改成了单引号，不严格的场合不加引号也是可以的，但是不建议这样做，因为浏览器会误解某些不加引号的属性。

下一步，我们显示商品信息到单元格中。不要跟在我后面完成，而且在我描述完要实现的功能后，尝试着自己实现，最后再看我的代码。

图片我放到了 img 目录中，数据库存放路径的时候使用了相对路径 img/abc.jpg，这个相对路径指向 webapps/365buy/img 目录，所以要将图片放到这个目录中，另外提一下，如果这样写路径/img/abc.jpg，那么访问的是 webapps/img。具体代码如下。

```
package com.wy;

import javax.servlet.http.*;
import java.io.*;
import java.sql.*;
```

```
public class Products extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse
response) {
        try {
            // 设置中文显示
            response.setContentType("text/html;charset=gb2312");

            // 准备输出流
            PrintWriter out = response.getWriter();

            // 准备数据库连接
            Class.forName("org.gjt.mm.mysql.Driver");
            Connection cn = DriverManager.getConnection(
               "jdbc:mysql://127.0.0.1:3306/360buy", "root", "123456");

            // 查询数据
            Statement st = cn.createStatement();
            ResultSet rs = st.executeQuery(
                "select img , name , price , gift , down , info from
products");

            // 输出 HTML 基本标记
            out.println("<html>");
            out.println("<body>");

            out.println("<style>");
            out.println(".p1 .p2{"};
            out.println("display:inline-block;");
            out.println("width:28px;");
            out.println("height:16px;");
            out.println("margin:0 2px;");
            out.println("overflow:hidden;");
            out.println("background-repeat:no-repeat;");
            out.println("line-height:100px;");
            out.println("font-size:0;");
            out.println("margin-bottom:-3px;");
            out.println(")");

            out.println(".name{"};
            out.println("font-size:12px;");
            out.println("padding:0px 20px 0px 20px;");
            out.println(")");

            out.println(".price{"};
            out.println("color:red;");
            out.println("font-size:15px;");
            out.println("font-weight:bold;");
```

```
            out.println(")");

            out.println("</style>");
            out.println("<table width='100%'>");

            int count = 0;
            while (rs.next()) {
                if (count % 4 == 0) {// 每四个商品一行
                    out.println("<tr>");
                }
                out.println("<td align='center'>");

                // 显示图片
out.println("<div><img src='");
out.println(changCode(rs.getString(1)));
out.println("'></div>");

// 显示商品名称
                out.println("<div class='name'>");
                out.println("<a href='' title='");
                out.println(changCode(rs.getString(6))+"'>");//提示信息
                out.println(changCode(rs.getString(2))+"</a>");
                out.println("</div>");

                // 显示价格
                out.println("<div class='price'>&yen;");
                out.println(rs.getDouble(3) + "</div>");

                //评价数量
                out.println("已有 0");
                out.println("人评价");

                if(rs.getInt(4)!=0){
                    out.println("<a class='p1' title='购买本商品送赠品'><img
src='img/gift.png'></a>");
                }
                if(rs.getString(5).equals("y")){
                    out.println("<a class='p2' title='本商品正在降价销售中
  '><img src='img/down.png'></a>");
                }

                //按钮区
                out.println("<div>");
                out.println("<input type='button' value='购买'>");
                out.println("<input type='button' value='关注'>");
                out.println("<input type='button' value='对比'>");
                out.println("</div>");
                out.println("</td>");
```

```
                // 请理解标记结束的逻辑，为什么要先++再判断
                count++;
                if (count % 4 == 0) {
                    out.println("</tr>");
                }
            }
            out.println("</tr>");
            out.println("</table>");
            out.println("</body>");
            out.println("</html>");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

//将 ISO-8859-1 编码转换成 GB2312，因为 MySQL 默认为 ISO-8859-1 编码，这样汉字
  //是乱码
    private String changCode(String s) throws UnsupportedEncodingException{
        if(s==null) {
return "";
}
return new String(s.getBytes("ISO-8859-1") , "GB2312");
    }
}
```

以上的代码我不加黑了，这些代码编写出来相当的不容易，几乎是一个个部分的反复调试，让人头疼的是，每修改那怕是一个字符，都要复制文件，重启 Tomcat。

我不得不在 catch 语句中加入对异常的打印，我们同样在这个过程中会犯错误，打印异常依然是重要手段，如果页面上展示的内容少了一些，甚至没有内容，打开源文件也发现 HTML 代码突然不正常的中断了，那么通常是出现异常了，打印异常看来是最直接的办法，但是你要清楚，现在打印的异常信息将出现在 Tomcat 的控制台窗体中，事实上 Tomcat 是可以没有黑色控制台启动的，但是我认为对于程序员来说，这个控制台太重要了，可以在这里打印出异常。

在很多细节上我并没有做到和参考的网页一模一样的程度，有些 JavaScript 的效果也没能加入，但是至少已经很像了，我很抱歉，我传递了一个错误的价值观，程序员要追求的不是差不多，而是尽可能的完美，想想苹果的产品为什么会如此成功吧。超越我，完美地实现这个网页，原版的商品名称是没有下划线的，当鼠标放到商品名称上面的时候，下划线出现了，还有很多这样的效果，甚至在原版的基础上，你也可以想到更好的效果。

我们体会一下，我们在做什么。这个过程是心中装着 HTML，手里写着 Java 代码，因为 HTML 代码是通过 Java 的输出语句产生的，所以编写网页的过程比单纯写 HTML 要难上两倍，加上部署的烦琐过程，CSS 部分尤其显得痛苦，其实 CSS 完全可以分离出去，我们不得不承受这样烦琐的代码，是为了利用 Java 访问数据库的能力，而 CSS 没有这样的要求。

我不单独拿出篇幅来展示这个变化，你有能力修改代码来达到这样的目的。

3.4.3　查询评论数量

在上面的代码中评论的数量是假的，那个 0 是我直接写上去的，我们需要用代码到评论的表中具体查一下有多少评论，我设计的评论表只有两列，一列是商品编号，第二列是评论，因为评论是和具体的某个商品关联的，得到某个商品的评论数，就是发出一个有 count 函数的 SQL 语句，条件是商品编号。

在刚才的代码中，因为没有直接用到商品编号，所以查询语句也就没有选择 pid 这个字段，现在要加上去。

```
ResultSet rs = st.executeQuery(
    "select img , name , price , gift , down , info , pid from
products");
```

现在来看查询评论数是每次循环得到一个商品的时候，就要进行一次，这种情况下，我需要 PreparedStatement，过去我们认为 PreparedStatement 的主要好处就是不需要进行不舒服的字符串连接，其实 PreparedStatement 的设计初衷是为了能够进行批次的查询，在我们这个任务中，就体现了批次查询的作用。我们在循环的外面准备 PreparedStatement。

```
    PreparedStatement ps = cn.prepareStatement("select count(pid) from
comment where pid=?");
```

然后在循环里面需要显示评论数的位置编写如下代码。

```
ps.setInt(1, rs.getInt(7));
ResultSet crs = ps.executeQuery();
crs.next();
out.println("已有");
out.println(crs.getInt(1));
out.println("人评价");
```

如果没有 PreparedStatement，我们就要在每次获取评论数的位置，准备查询语句，而现在我们就准备一次，只要每次赋予不同的值，然后查询就好了。

3.4.4　分离数据库连接

在这个任务中，将获取数据库连接的代码分离出去的需求并不强烈，因为我们只有一个点上需要连接数据库，但是如果从整个购物网站来看，会有很多地方需要连接数据库，分离这段代码的真正好处是为了在数据库改变的时候，程序更容易维护，事实上几乎所有的软件项目，数据库的连接都是使用工厂提供的。

我们创建一个新的类 DataBase，这个类有个静态的返回值是 Connection 的方法。既然我们要做这样的分离，那么我们就做的彻底一些，将数据库连接的字符串放到配置文件中，现在提到配置文件，应该想到 XML，正常来想应该定义 DTD，然后编写 XML，最后在 Java 中用 DOM 读 XML 中的内容，不过在这里我们到不用费劲的从头来，系统中不是已经有 XML 文件了吗？

我指的就是那个 web.xml，我们就借用 web.xml 好了，可是 web.xml 是用来配置 Servlet 的，要想和 web.xml 打交道，就得是个 Servlet，干脆我们将 DataBase 写成 Servlet 再配置到 web.xml 中去，这样我们就可以使用 Servlet 初始化参数了。下面是 web.xml 中的代码。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  version="3.0"
  metadata-complete="true">

  <servlet>
  <servlet-name>show</servlet-name>
  <servlet-class>com.wy.Products</servlet-class>
  </servlet>

  <servlet>
<servlet-name>data</servlet-name>
  <servlet-class>com.wy.DataBase</servlet-class>
  <init-param>
     <param-name>driver</param-name>
     <param-value>org.gjt.mm.mysql.Driver</param-value>
  </init-param>
  <init-param>
     <param-name>url</param-name>
     <param-value>jdbc:mysql://127.0.0.1:3306/360buy
     </param-value>
  </init-param>
  <init-param>
     <param-name>user</param-name>
     <param-value>root</param-value>
  </init-param>
  <init-param>
     <param-name>pass</param-name>
     <param-value>123456</param-value>
  </init-param>
  </servlet>

  <servlet-mapping>
  <servlet-name>show</servlet-name>
  <url-pattern>/show.bin</url-pattern>
  </servlet-mapping>

</web-app>
```

我们添加了一个 servlet 元素，前面都是一样的，但是在这个 servlet 中添加了 4 个 init-param 元素，其中有名字和值，看得出来，我就是在 init-param 中填写了数据库连接所需的字符串。

我并没有为这个叫做 data 的 Servlet 提供对应的 servlet-mapping，理由很简单，这个 Servlet 并不需要浏览器来访问。

下面是 DataBase 类的代码。

```
package com.wy;

import javax.servlet.http.*;
import java.sql.*;

public class DataBase extends HttpServlet{
    private static String driver;
    private static String url;
    private static String user;
    private static String pass;
    public void init(){
        driver = this.getInitParameter("driver");
        url = this.getInitParameter("url");
        user = this.getInitParameter("user");
        pass = this.getInitParameter("pass");
    }
    public static Connection getConnection() {
        Connection cn = null;
        try{
            Class.forName(driver);
            cn = DriverManager.getConnection(url , user , pass);
        }catch(Exception e){}
        return cn;
    }
}
```

事实上我们有三种途径实现 Servlet，对于这个程序，选择那一种都无所谓，因为我们只是让它随着 Servlet 启动就可以了，并不实现 Servlet 主要的功能，这里我继承了 HttpServlet，因为这样做不需要提供 service 方法。

在类中我们看到有两个方法，getConnection 方法不用说，这是我们要实现的方法，init 方法并不是我编造的，这就是 Servlet 规范中的启动方法，在这个方法中，我们能够使用 getInitParameter 方法，就是用这个方法读取 web.xml 中的 init-param 元素，参数是名字，返回值是 XML 中的值。

现在要将 Products 类中连接数据库的语句。

```
// 准备数据库连接
Class.forName("org.gjt.mm.mysql.Driver");
Connection cn = DriverManager.getConnection(
    "jdbc:mysql://127.0.0.1:3306/360buy", "root", "123456");
```

替换成：

```
Connection = DataBase.getConnection();
```

然后我们部署、测试一下，你会发现页面打不开的，你试一下问题出在哪里。

我们在 Products 中打印异常，会发现报告变量 cn 的空指针异常，这说明没有得到 Connection，那么问题就出在 DataBase 类中，再打印 DataBase.getConnection 方法中的异常，发现字符串都是空值。

我告诉你原因吧，因为这个 Servlet 根本就没运行起来，还记得我们没有提供 servlet-mapping 吗？Tomcat 以为你这个 Servlet 不需要运行呢。现在加一句话就能强制运行这个 Servlet 了。具体代码如下。

```
<servlet>
    <servlet-name>data</servlet-name>
    <servlet-class>com.wy.DataBase</servlet-class>
    <init-param>
        <param-name>driver</param-name>
        <param-value>org.gjt.mm.mysql.Driver</param-value>
    </init-param>
    <init-param>
        <param-name>url</param-name>
        <param-value>jdbc:mysql://127.0.0.1:3306/360buy</param-value>
    </init-param>
    <init-param>
        <param-name>user</param-name>
        <param-value>root</param-value>
    </init-param>
    <init-param>
        <param-name>pass</param-name>
        <param-value>123456</param-value>
    </init-param>
    <load-on-startup>10</load-on-startup>
</servlet>
```

这句话指明该 Servlet 在第 10 个位置启动，这个位置非常靠前了，因为 Tomcat 本身还有很多要启动的 Servlet，原本设计这个装载顺序是为了协调 Servlet 之间的依赖关系，有时一个 Servlet 启动的时候需要其他 Servlet 先启动，我们现在的程序就有这样的需求，这样就要设置这个 XML 元素。

3.4.5 分页显示

分页显示是项目比较多的时候不可避免的一个功能，由于我们现在数据比较少，所以我设定每页显示 8 个商品。先自己想一下实现的思路，如果翻到第二页，就意味着要显示从第 9 个商品到第 16 个商品，那么我们能使用 SQL 语句得到这个范围的商品吗？除了在 Oracle 中使用一个十分高级的技巧可以得到记录中的一段外，我不知道其他数据库可以单纯使用查询语句实现类似的查询，你可以研究一下这个课题，有的数据库可以用存储过程实现这个功能，大多数数据库都可以取得前几条记录，但是这不管用。

我们先将这个思考放到一边，第一步我们就实现第一个页面，在这个页面中，我们显示最前面的 8 个商品，然后在页面的下面显示页码，你可以独自完成这一步。具体代码如下。

```
package com.wy;

import javax.servlet.http.*;
import java.io.*;
import java.sql.*;

public class Products extends HttpServlet {
   public void doGet(HttpServletRequest request,
         HttpServletResponse response) {
      try {
         // 设置中文显示
         response.setContentType("text/html;charset=gb2312");

         // 准备输出流
         PrintWriter out = response.getWriter();

         // 准备数据库连接
         Connection cn = DataBase.getConnection();

         // 查询数据
         Statement st = cn.createStatement();
         ResultSet rs = st.executeQuery(
   "select img, name, price, gift, down, info, pid from products");

         // 输出 HTML 基本标记
         out.println("<html>");
         out.println("<body>");
         out.println("<link rel=stylesheet
            href='show.css' type='text/css'>");
         out.println("<table width='100%'>");

         PreparedStatement ps = cn.prepareStatement(
            "select count(pid) from comment where pid=?");
         for(int i = 0; i < 8; i ++){
            rs.next();
            if (i % 4 == 0) {// 每四个商品一行
               out.println("<tr>");
            }
            out.println("<td align='center'>");

            // 显示图片
            out.println("<div><img src='");
            out.println(changCode(rs.getString(1)));
            out.println("'></div>");
```

```
                // 显示商品名称
                out.println("<div class='name'>");
                out.println("<a href='' title='");
                out.println(changCode(rs.getString(6))+"'>");
                //提示信息
                out.println(changCode(rs.getString(2))+"</a>");
                out.println("</div>");

                // 显示价格
out.println("<div class='price'>&yen;");
                out.println(rs.getDouble(3) + "</div>");

                //评价数量
                ps.setInt(1, rs.getInt(7));
                ResultSet crs = ps.executeQuery();
                crs.next();
                out.println("已有");
                out.println(crs.getInt(1));
                out.println("人评价");

                if(rs.getInt(4)!=0){
                    out.println("<a class='p1' title='购买本商品送赠品
'><img src='img/gift.png'></a>");
                }
                if(rs.getString(5).equals("y")){
                    out.println("<a class='p2' title='本商品正在降价销售中
'><img src='img/down.png'></a>");
                }

                //按钮区
                out.println("<div>");
                out.println("<input type='button' value='购买'>");
                out.println("<input type='button' value='关注'>");
                out.println("<input type='button' value='对比'>");
                out.println("</div>");
                out.println("</td>");

                // 请理解标记结束的逻辑，为什么要先++再判断
                if ((i+1) % 4 == 0) {
                    out.println("</tr>");
                }
            }
            out.println("</tr>");
            out.println("</table>");

            //显示页码
            //获得总行数
            rs = st.executeQuery(
```

```
                "select count(pid) from products");
            rs.next();
            int rowCount = rs.getInt(1);
            //计算总页数
            int pageCount = 0;
            if(rowCount%8==0){
                pageCount = rowCount/8;
            }else {
                pageCount = rowCount/8 + 1;
            }
            //显示
            for(int i = 1 ; i <= pageCount ; i ++){
                out.println("<a href=''>"+i+"</a>");
            }
            out.println("</body>");
            out.println("</html>");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    private String changCode(String s) throws UnsupportedEncoding Exception{
        if(s==null) {
            return "";
        }
        return new String(s.getBytes("ISO-8859-1"), "GB2312");
    }
}
```

上面的代码是在之前代码基础上修改得到的，我们的任务是显示第一页，显示第一页的 8 个商品相当容易，只需要将原本的 while 循环改成循环 8 次的 for 循环就好了，rs.next()还是需要的。由于 for 循环中有自增量 i，所以我判断是否加上 tr 标记的 count 变量不再需要了。

再来看显示页码，我们要知道一共有多少页，那么首先要知道一共有多少个商品，这个发出一条带 count 的查询语句就能够得到，知道总的商品数，还知道每页显示 8 个商品，那么除一下就知道总页数了，但是有个小问题，就是能否除得开，要知道如果除不开，总页数会少一页，因为整数之间的除法是舍尾操作，所以要判断一下，如果除不开，就在结果上加一，否则不加。

不单单要显示页码，当用户单击页码的时候需要跳到下一页，毫无疑问，每个页码都是一个超链接，问题是应该链到那里，总不能有 100 页，就写 100 个 Servlet 吧，再说每一页的逻辑是一样的，看来应该链接到同一个 Servlet，只不过要告诉它要显示的是那一页，这个 Servlet 就是 Products，就是自己。

那么如何传递页码信息呢？我们学过的向下一个页码传递信息，是利用 form 表单，再想想当提交这个表单后，用户名和密码出现在什么地方？没错，如果不是 POST 方式的话，信息将出现在地址栏的问号后面，我们完全可以伪装这个信息传递，将页码的输出改写如下。

```
for(int i = 1; i <= pageCount; i ++){
    out.println("<a href='show.bin?page="+i+"'>"+i+"</a>");
}
```

那么按照流程，下一个任务是在程序的开始接收传递过来的页码。

```
//接收当前页码
int nowPage = Integer.parseInt(request.getParameter("page"));
```

无论是什么数据类型，在页面之间传递都是以字符串传递的，所以使用的时候页码要转换成数字。

再次来到我们开始就思考的问题上，如果是第二页要显示，程序该如何实现，前提是 SQL 语句无法提供第二页所需的信息，那么 SQL 语句将提供给我全部的信息，只剩下最笨的办法，在显示前，将不需要的商品跳过去。

```
//跳过不需要的商品
for(int i = 0; i < (nowPage-1)*8; i ++){
    rs.next();
}
```

现在测试一下是否能够实现分页。大部分情况下没问题，但是有两个地方有问题。

一是到了最后一页，会发现页码消失，思考一下问题可能出在那里，如果没有思路，就看一下源文件，你会发现 HTML 代码突然结束了，后面的一部分代码没有了，你知道这是发生什么事情了吧？出异常了，那么打印异常看看，原因是在最后一页，不够 8 个商品了，但是 rs.next(); 还会执行，游标的指针指不到数据上，这样取值的时候就会发生异常，解决办法是进行判断，好在 rs.next()的返回值是 boolean 值，所以判断这个返回值就可以了，我们将 if 语句加上去。

```
if(rs.next()) {
    //循环体中所有的语句。
}
```

第二个问题是，当第一次访问 http://127.0.0.1:8080/365buy/show.bin 的时候，什么都看不见，不用说，还是出异常了，打印异常后知道，是接收 page 值然后转换成数字的时候出了问题，因为直接访问的时候，没有后面？page=，这样取值得到的是 null，将 null 转变成数字就会出异常，看来我们还是要判断一下。

```
//接收当前页码
String page = request.getParameter("page");
int nowPage = 1;
if(page!=null){
    nowPage = Integer.parseInt(page);
}
```

如果没有得到 page 值，那么就显示默认的第一页。问题解决了，但是如果有人恶意的输入 page 值怎么办？http://127.0.0.1:8080/365buy/show.bin?page=abc，这样还会出异常，事实上很多黑客，就是通过伪装的地址栏访问，注入恶意代码来攻击网站，看来只判断是否为 null 还不行，但是可能的问题变化多端，如何避免呢？其实我们就是想要数字，判断是否是数字就好了，

我给出这样的代码。

```
//接收当前页码
int nowPage = 1;
try{
    nowPage = Integer.parseInt(request.getParameter("page"));
}catch(Exception ee){}
```

通过 try catch 语句来解决这个问题，如果出现异常，那么 nowPage 的值将是声明时给的默认值 1，恰好是显示第一页。这是对 try catch 巧妙地应用，你可以体会一下。

3.4.6　在每个页面上都显示用户名

我们看到京东的网站上，如果你成功的登录了，那么每个页面上方都会显示这个用户名，我实现的时候忽略上面漂亮的效果，就是完成显示“你好，”后面加上用户名。

先要将上一个部分用户登录和这个部分的商品展示两个程序融合到一起，包括数据库的表、login.html、Loginc 类和 web.xml 都需要调整。我不介绍具体的调整步骤了，你有能力完成这样的任务。

已知的办法是通过伪装地址栏不断地向下一个页面传递用户名，这必定会很麻烦，能不能将用户名存放在一个地方，每个页面都去那个地方取用户名的值，这个想法实现起来还真不容易，因为 HTTP 协议访问的模式是断续的，不是一直稳定连接的，如果是稳定连接的，就类似于 QQ 服务器，我们可以启动一个专门的线程来为一个用户服务，那么在线程中声明的变量，这个用户无论在什么时候都能访问到。

而现在是浏览器请求一个页面的时候，网络连接上，得到 HTML 文件后，网络就可以断开了，下次请求再进行连接就好了，在浏览器不访问服务器的时候，服务器根本不知道这个浏览器是否还在，对这个浏览器服务的线程也不会等待着，通常会为另外的浏览器请求服务，也就是说服务器端的线程并不是为一个特定的浏览器服务，这样我们就不能利用线程的作用域保存信息了，但是可以想象保存信息的需求应该是相当普遍的，为此 Tomcat 提供了一个叫做 session 的集合，用来存放跨越网页的信息。

要说 session 实现起来并不容易，一方面它的作用域要能够跨越不同的网页，另一方面它要能够区分不同的用户，就是说你无论如何访问这个网站，得到的 session 都将是一个属于你的，而别人不能看见你的 session，这就要求 Tomcat 能够识别不同的人，人是识别不了，session 识别的是不同的浏览器，还记得我们做假 Tomcat 的时候，得到了一个时间戳吗？Tomcat 就是通过匹配时间戳来决定给你开放哪个 session 的。

我们总结一下，session 是 Tomcat 提供并维护的。session 通过时间戳来识别不同的浏览器，确保一个 session 只能由你来使用。session 能够跨越页面。session 是个集合，这种情况下集合最实用，能够装任何对象，而且不限数量。

现在要完成这个阶段的任务，只要将用户名放到 session 中就可以了。找回用户登录的处理

程序 Loginc.java，我们要在用户验证成功后将用户名存放到 session 中。具体代码如下。

```
package com.wy;

import javax.servlet.http.*;
import java.sql.*;

public class Loginc extends HttpServlet {
   public void doPost(HttpServletRequest request, HttpServletResponse
response) {
       try {
          response.setContentType("text/html;charset=GB2312");

          String user = request.getParameter("username");
          String pass = request.getParameter("password");

          Connection cn = DataBase.getConnection();
          PreparedStatement ps = cn.prepareStatement("select * from
user where username=? and password=?");
          ps.setString(1, user);
          ps.setString(2, pass);
          ResultSet rs = ps.executeQuery();
          if (rs.next()) {
             HttpSession session = request.getSession();
             session.setAttribute("user", user);
             response.sendRedirect("show.bin");
          } else {
             response.sendRedirect("reg.bin");
          }
       } catch (Exception e) {}
   }
}
```

你能看到 session 不是 new 出来的，而是获取的，因为创建 session 的是 Tomcat，不是你写的程序。Session 是两列的 Map，但是经过了适当的改造，前面的 key 被规定为 String 类型。

我要在商品展示的页面读取 session 中的 user，如果读到了就显示出来。我不展示全部的代码了，就展示显示用户名的代码。

```
//显示用户名
HttpSession session = request.getSession();
String user = (String)session.getAttribute("user");
if(user==null){
   out.println("<a href='login.html'>请登录</a>");
}else {
   out.println("欢迎，"+user);
}
```

3.5　用户注册

我们终于有能力编写用户注册代码了，简单理解会认为我们需要一个 reg.html，提交后有一个 regc.bin 的 Servlet 验证输入的信息，如果信息没有问题就添加到数据库中，但是由于有了验证码，情况变得复杂了，为了学习如何产生验证码，我们索性整个完成用户注册吧。

在学习 HTML 时，我们已经实现了京东商城用户注册的效果，所以这里我还是不纠缠界面的美观和 JavaScript 的功能，写一个最简单的注册页面，重点在验证码。

和之前我们所学习的所有技术都不同，之前的技术对于你来说就是知道和不知道的问题，但是验证码是更高深的技术，最初的网站是没有验证码的，这样一些黑客就利用程序自动的注册大量的垃圾用户，并且用这些僵尸用户进行操作。后来人们发现有些敏感的用户登录，也会有人用程序不断地尝试密码，因此渐渐地很多网站的用户登录也使用了验证码。

验证码的原理是，在服务器端产生一个随机数，也可能是字母数字序列，然后产生这个随机数的图片，将图片显示到网页上，用户需要识别出来这个网页的内容，填入验证码的输入框中，连同注册或登录信息一同发到服务器上，服务器接收到以后，比对最初产生的随机数，如果一致才进行操作。

这样实现验证码就包含了三个小任务，产生随机数、生成图片和比对，难点在于生成图片，比对要求随机数产生后，不能丢弃，而是要存放在 session 中。

很快黑客们就找到了 OCR（文字识别软件），对网页上的图片进行自动地识别，网站的编写者再次遭遇挑战，为了对抗 OCR，程序员在生成的图片上面加入了干扰线，然后 OCR 解决了干扰线，现在的网站编写者尝试着让文字变形，同时还要兼顾着不能变换到用户都看不懂。

验证码的技术还在发展之中，因为 OCR 的能力也在发展，所以我提供的程序必定会过时，你需要自己探索更好的算法。

首先我们准备 reg.html 的代码，其中显示验证码图片的位置使用 img 标记，src 属性指向生成图片的 Servlet。具体代码如下。

```
<html>
<body>
<h1>用户注册</h1>
<form action="regc.bin" method="POST">
用户名：<input type="text" name="user"/><br/>
密码：<input type="password" name="pass"/><br/>
确认密码：<input type="password" name="rpass"/><br/>
邮箱：<input type="text" name="email"/><br/>
验证码：<input type="text" name="vali"/>
<img src="vali.bin" width="80" height="25"/>
看不清？<a href="">换一个</a>
</form>
```

```
</body>
</html>
```

你可以在此基础上将这个页面美化一下，并且使用 JavaScript 来实现一些效果，如果没问题，我们将注意力放在显示验证码这个部分。

3.5.1 生成验证码图片

现在我们编写显示验证码图片的 Servlet，我们先来实现第一步，生成随机字符串，思路是在 init 中准备一个 62 位的字符串数组，里面存放着 10 个数字、26 个小写字母和 26 个大写字母。在需要生成随机字符串代码处，用 0 到 62 之间的随机数到数组中取值，将取得的每一位追加到空字符串中，循环 4 次就得到了 4 位随机生成的字符串。你可以使用其他的算法完成这个任务。具体代码如下。

```
package com.wy;

import javax.imageio.ImageIO;
import javax.servlet.http.*;
import java.io.*;
import java.awt.*;
import java.awt.image.*;

public class Vali extends HttpServlet{
    String ver[] = new String[62];
    public void init(){
        for(int i = 0; i < 10; i ++){
            ver[i] = new Integer(i).toString();
        }
        for(int i = 0; i < 26; i ++){
            ver[i+10] = new Character((char)(97+i)).toString();
        }
        for(int i = 0; i < 26; i ++){
            ver[i+36] = new Character((char)(65+i)).toString();

        }
    }
    //生成随机数字和字母
    private String getValidateCode(){
        StringBuffer vali = new StringBuffer();
        for(int i = 0; i < 4; i ++){
            vali.append(ver[(int)(Math.random()*36)]);
        }
        return vali.toString();
    }

    public void doGet(HttpServletRequest request,
HttpServletResponse response){
```

```
        try{
        }catch(Exception e){}
    }
}
```

第二步获得绘图的能力，回顾一下，过去我们绘图是因为继承了 Panel，并且重写了 paint 方法，系统通过方法的参数提供了绘图能力给我，毫无疑问现在的机制完全不同，我们要将图片画到浏览器上，而服务器和浏览器打交道的唯一方式就是 IO 流，所以我们要在服务器代码中产生图片，然后用输出流送到客户端。具体代码如下。

```
    public void doGet(HttpServletRequest request,
            HttpServletResponse response){
        try{
            // 设置图像输出
            response.setContentType("image/jpeg");
            // 获取输出流
            OutputStream os = response.getOutputStream();
            // 在内存中准备一个 image
            BufferedImage image = new BufferedImage(50 , 20 ,
BufferedImage.TYPE_INT_RGB);
            Graphics g = image.getGraphics();

            // 绘图
            g.setColor(new Color(200, 200, 200));
            g.fillRect(0, 0, 50, 20);
            g.dispose();

            // 以 image 的形式输出
            ImageIO.write(image, "JPEG", os);
        }catch(Exception e){}
}
```

在产生图片的这一步，图片和任何硬件设备都没有关联，就是在内存中产生这个图片，所以我直接 new 一个图片对象。

从图片对象得到“画笔”，这样就来到我们熟悉的绘图代码，当绘图的工作完成，我们使用 ImageIO 类的成员方法 write 将图片对象，以 JPEG 的形式输出到 os 输出流去。

测试一下访问 reg.html 能够看见一个灰色的小矩形，这中间要部署这个 Servlet，并且修改 web.xml，我就不啰嗦了。

现在我们将随机字符串画上去。具体代码如下。

```
// 绘图
g.setColor(new Color(200, 200, 200));
g.fillRect(0, 0, 50, 20);

// 生成并绘制随机字符串
String vali = "";
```

```
for (int i = 0; i < 4; i++) {
    String v = (ver[(int) (Math.random()*36)]);
    vali += v;
    g.setColor(new Color((int)(Math.random()*150),(int)(Math.random()*
150) , (int)(Math.random()*150)));
    g.drawString(v, 8*i+10, 15);
}

g.dispose();
```

为了让每个字符都有一个随机的颜色，所以我将前面生成随机字符串的方法去掉，其中的代码放到生成并绘制随机字符串的位置，在设置随机的颜色问题上，我没有选择 0 到 255 这样的范围区间，而是到 150，因为矩形区域的底色是 200，这样能够确保显示的字符比底色深，用户能够看见。

现在重新部署测试，网页上就有随机的验证码了，每次刷新浏览器，都会得到一个不同的验证码。

显示出来后，我们要将生成的随机字符串存入 session，这样在 regc.bin 进行注册处理的代码中，就可以取用户的输入和 session 中预存的值进行比对了。

```
// 保存随机生成的验证码
HttpSession session = request.getSession();
session.setAttribute("vali", vali);
```

3.5.2 绘制干扰线

我们进一步研究一下如何对抗 OCR 的自动识别，先在图片中加入干扰线。具体代码如下。

```
// 绘制干扰线
g.setColor(new Color(150 , 150 ,150));
for(int i = 0; i < 20; i++){
    int x1 = (int)(Math.random()*50);
    int y1 = (int)(Math.random()*20);
    int x2 = (int)(Math.random()*50);
    int y2 = (int)(Math.random()*20);
    g.drawLine(x1, y1, x2, y2);
}
```

这段代码放到显示随机字符串的上面，具体代码如下。

```
package com.wy;

import javax.imageio.ImageIO;
import javax.servlet.http.*;
import java.io.*;
import java.awt.*;
import java.awt.image.*;
```

```
public class Vali extends HttpServlet {
    String ver[] = new String[62];

    public void init() {
        for (int i = 0; i < 10; i++) {
            ver[i] = new Integer(i).toString();
        }
        for (int i = 0; i < 26; i++) {
            ver[i + 10] = new Character((char) (97 + i)).toString();
        }
        for (int i = 0; i < 26; i++) {
            ver[i + 36] = new Character((char) (65 + i)).toString();

        }
    }

    public void doGet(HttpServletRequest request,
            HttpServletResponse response) {
        try {
            // 设置图像输出
            response.setContentType("image/jpeg");
            // 获取输出流
            OutputStream os = response.getOutputStream();
            // 在内存中准备一个 image
            BufferedImage image = new BufferedImage(50, 20,
                    BufferedImage.TYPE_INT_RGB);
            Graphics g = image.getGraphics();

            // 绘图
            g.setColor(new Color(200, 200, 200));
            g.fillRect(0, 0, 50, 20);

            // 绘制干扰线
            g.setColor(new Color(150, 150, 150));
            for(int i = 0; i < 20; i++){
                int x1 = (int)(Math.random()*50);
                int y1 = (int)(Math.random()*20);
                int x2 = (int)(Math.random()*50);
                int y2 = (int)(Math.random()*20);
                g.drawLine(x1, y1, x2, y2);
            }

            // 生成并绘制随机字符串
            String vali = "";
            for (int i = 0; i < 4; i++) {
```

```
                String v = (ver[(int) (Math.random() * 36)]);
                vali += v;
                g.setColor(new Color((int)(Math.random()*150),
(int)(Math.random()*150), (int)(Math.random()*150)));
                g.drawString(v, 8*i+10, 15);
            }

            // 保存随机生成的验证码
            HttpSession session = request.getSession();
            session.setAttribute("vali", vali);

            g.dispose();

            // 以 image 的形式输出
            ImageIO.write(image, "JPEG", os);
        } catch (Exception e) {}
    }
}
```

显示效果如图 3-21 所示。

图 3-21

这样的验证码还是比较弱的，现在出现了很多字符显示的变换和其他类型的验证码，字符显示变换常见的有这样几种类型，字体和大小随机变化，显示位置变化，字符变形显示，前两种变化比较好实现，字符变形显示需要动用三角函数。还有回答问题的验证码，找图片的验证码等形式。

3.5.3 更新验证码

用户单击图片或是单击“换一个”链接时，应该更换图片，这样我们需要重新运行一遍 vali.bin，如何重新运行是个问题，如果我们发出一个新的对 vali.bin 的请求，浏览器从效率角度考虑会使用已有的缓存，除非请求的 URL 发生了改变，看代码。

```
<img src="vali.bin" id="im" onClick="this.src=this.src+'?'+Math.
random();" width="80" height="25"/>
```

看不清？换一个

关键的代码是：this.src=this.src+'?'+Math.random();，我们还是第一个在 JavaScript 中用到

this，不过不难理解，就是当前的这个对象，this.src=this.src，事实上就是对当前的属性进行一次重新的请求，但是这样的语句会被浏览器误认为是没有必要的语句，所以在后面加上？和一个随机数，这样请求就不同了，vali.bin 被迫重新工作一次，还有些人取时间值放到后面，道理是一样的。

3.5.4　注册处理程序

现在我们来实现注册处理程序，任务分成四个部分，得到用户的输入，识别验证码，验证用户是否重名，存入数据库。其中输入有效性以及两次密码的一致性验证由 JavaScript 完成，不在我们要实现的代码功能中。

第一步得到用户输入的代码非常简单，我同时提供识别验证码的具体代码。

```
package com.wy;

import javax.servlet.http.*;

public class Regc extends HttpServlet {
    public void doPost(HttpServletRequest request,
            HttpServletResponse response){
        try{
            String user = request.getParameter("user");
            String pass = request.getParameter("pass");
            String email = request.getParameter("email");
            String vali = request.getParameter("vali");

            //识别验证码
            HttpSession session = request.getSession();
            String valiCode = (String)session.getAttribute("vali");

            if((vali==null)||(vali.equals(""))){
                session.setAttribute("mess", "请输入验证码!");
                response.sendRedirect("reg.bin");
                return ;
            }
            if(!(valiCode.equals(vali))){
                session.setAttribute("mess", "验证码不正确，请重新输入!");
                response.sendRedirect("reg.bin");
            }
        }catch(Exception e){}
    }
}
```

我们看到在识别验证码的逻辑中，可能存在两种错误，用户没有输入验证码或是验证码不正确，正常情况下，无论发生什么样的验证码错误，都应该将网页跳转回用户注册页面，并且在页面上显示错误信息，这就有问题了，我们之前的代码，用户注册页面是个 HTML 文件，到目前

为止 HTML 文件没法接受服务器发送过来的信息，并且显示出来，除非用户注册页面是个 Servlet，所以我们跳转到 reg.bin，然后要改造成 Servlet。

在新的 Reg 的 Servlet 中，检测并用红字显示 mess 内容，我就放到标题的下面，你可以改造使其处于验证码的附近。具体代码如下。

```
package com.wy;

import javax.servlet.http.*;
import java.io.*;

public class Reg extends HttpServlet{
    public void doGet(HttpServletRequest request,
            HttpServletResponse response){
        try{
            // 设置中文显示
            response.setContentType("text/html;charset=gb2312");

            // 获取输出流
            PrintWriter out = response.getWriter();

            // 输出 HTML
            out.println("<html>");
            out.println("<body>");
            out.println("<h1>用户注册</h1>");
            out.println("<form action='regc.bin' method='POST'>");
            out.println("用户名：<input type='text' name='user'/> <br/>");
            out.println("密码：");
            out.println("<input type='password' name='pass'/><br/>");
            out.println("确认密码：");
            out.println("<input type='password' name='rpass'/><br/ >");
            out.println("邮箱：<input type='text' name='email'/><br/>");
            out.println("验证码：<input type='text' name='vali'>");
            out.println("<img src='vali.bin' width='50' height='25'>");
            out.println("看不清？<a href=''>换一张</a><br/>");
            out.println("<input type='submit' value='提交'>");
            out.println("</form>");
            out.println("</body>");
            out.println("</html>");
        }catch(Exception e){}
    }
}
```

将 reg.html 改造成为 Servlet 还有进一步的好处，这个我们回头再说。现在先集中精力完善这个 regc.bin，这一步判断用户名是否冲突。具体代码如下。

```
// 判断用户名是否冲突
Connection cn = DataBase.getConnection();
PreparedStatement ps = cn.prepareStatement(
```

```
    "select username from user where username=?");
ps.setString(1, user);
ResultSet rs = ps.executeQuery();
//有数据时错误，说明已经有了这个用户名了
if(rs.next()){
    session.setAttribute("mess", "该用户已经存在，请换一个名字!");
    response.sendRedirect("reg.bin");
    return;
}
```

这些都没有问题的话，我们将这个用户存入数据库的表中，在代码前，user 表需要做个调整，加入字段 email。

```
alter table user add column (email varchar(50)) ;
```

下面是 Servlet 的相关代码。

```
// 添加用户到数据库表中
ps = cn.prepareStatement(
    "insert into user(username, password, email) values(?,?,?)");
ps.setString(1, user);
ps.setString(2, pass);
ps.setString(3, email);

ps.executeUpdate();
```

我们设计的任务都完成了，但是编程到这，我们发现代码还没完成，用户成功注册后，我们的程序应该怎么办，如果就这样结束了，用户会得到一个空白的页面，这时会有几个答案，显示一个页面告诉用户注册成功了，或是跳转到登录页面，或者是自动进行登录，显示主页面，总之你要再做个事情，我选择替用户登录。具体代码如下。

```
//登陆到 show.bin
session.setAttribute("user", user);
response.sendRedirect("show.bin");
```

3.5.5　使用 AJAX 验证用户名是否冲突

在上面的代码中，我们已经能够验证用户名是否冲突了，但是这个实现并不高明，在这个过程中浏览器请求 reg.bin，用户输入信息，然后信息提交到服务器，由 regc.bin 来处理，regc.bin 发现用户名冲突，会再次传输 reg.bin 到浏览器，我们不得不经过一个来回将用户名发送到服务器，由服务器端代码来验证，因为只有这样才能和数据库打交道，同样的用户注册页面被传送了两次，仅仅因为两次的内容有一点点的不一样（多了错误信息）。

有没有办法不重复传送用户注册页面，而是在验证后，将错误信息加入到已经在浏览器上显示的页面呢？AJAX 便是拥有这样能力的技术，AJAX 的做法是实现异步的网页传送，这样说来，我们过去的做法就是同步的网页传送，同步的意思是这个网页一次性传送到浏览器，异步的就意味着我们只将一部分网页传送到浏览器，在需要的情况下，会传送另外的信息，浏览器可以将新接收的信息整合到已有的网页中。

那么，AJAX 是如何工作的呢？我们知道在访问网页的过程中，有浏览器和服务器两个主体，由于互联网的特殊性，这两个主体相当的独立，浏览器不关心服务器是什么软件支持的，而服务器也不关心浏览器是什么样的软件和版本，AJAX 就是利用这个特点对服务器端的程序实施了欺骗，这个欺骗是由 JavaScript 来进行的。

我们先来看传统的网络访问，如图 3-22 所示。

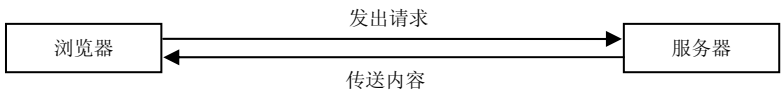


图 3-22

AJAX 的工作模式，如图 3-23 所示。

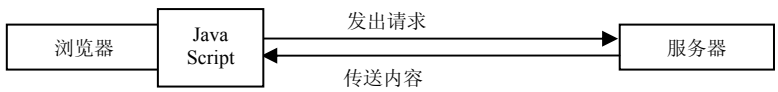


图 3-23

在 AJAX 中，JavaScript 代替了浏览器向服务器发出请求，服务器便会误认为 JavaScript 就是浏览器，于是将信息传送给 JavaScript，我们知道 JavaScript 有能力将信息写入网页中，所以我们就得到了异步的将信息塞入网页的效果。

那么，JavaScript 是如何伪装浏览器的呢？JavaScript 是个基于对象的语言，在不同浏览器上提供了不同的特定对象，这个对象就是伪装的浏览器，为了代码清晰，现在用 reg.html 来实现用户名判断的功能。具体代码如下。

```
<html>
<script>
// 创建 XmlHttpRequest 对象，不同的浏览器该对象的名称不同
function newObject(){
    if(window.XMLHttpRequest){// 非 IE 浏览器支持
        req = new XMLHttpRequest();
    }else if(window.ActiveXObject) {// 微软的 IE 浏览器支持
        req = new ActiveXObject("Microsoft.XMLHttp");
    }

    // 当对象的状态发生了变化，到 press 函数去处理，这很类似于 Java 的事件处理
    // 注意这是大小写敏感的语言，press 后面没有小括弧
    req.onreadystatechange=press;
    // 通过这句话将用户名发送到服务器端的验证程序
    req.open("GET", "valiUser.bin?user="+f1.user.value , true);
    req.send("");
}

// 服务器传送过来信息的响应程序
```

```
function press() {
    // readystate 的值，0 未初始化、1 正在装载、2 装载完毕、3 交互中、4 完成
    if(req.readyState==4) {
        // 还记得 200 OK 吗？我们不处理 404 或 500 的信息
        if(req.status==200) {
            alert(req.responseHTML);
        }
    }
}
</script>
<body>
    <h1>用户注册</h1>
    <form action="regc.bin" method="POST" name="f1">
        用户名：
        <input type="text" name="user" onBlur="newObject()"/><br/>
        密码：<input type="password" name="pass"/><br/>
        确认密码：<input type="password" name="rpass"/><br/>
        邮箱：<input type="text" name="email"/><br/>
        验证码：<input type="text" name="vali"/>
        <img src="vali.bin" width="80" height="25"/>
        看不清？<a href="">换一个</a>
    </form>
</body>
</html>
```

上面的 HTML 代码我给出了比较详细的注释，JavaScript 部分代码可以总结成三个部分，第一部分创建对象，第二部分设置对象，说明如果有信息传送过来，处理程序是什么，以及传送什么信息到服务器，第三部分判断对象的状态，接收信息。

通常接收了信息要进行处理，但是我们的代码先将信息显示处理，再确保能够使用 JavaScript 异步的获取服务器发送过来的信息后，我们再进行处理。

现在我们要写 ValiUser 的代码了。

```
package com.wy;

import javax.servlet.http.*;
import java.sql.*;
import java.io.*;

public class ValiUser extends HttpServlet{
    public void doGet(HttpServletRequest request,
            HttpServletResponse response){
        try{
            // 设置中文，获取输出流
            response.setContentType("text/html;charset=gb2312");
            PrintWriter out = response.getWriter();

            String user = request.getParameter("user");
```

```
            // 到数据库中验证
            Connection cn = DataBase.getConnection();
            PreparedStatement ps = cn.prepareStatement(
                "select username from user where username=?");
            ps.setString(1, user);
            ResultSet rs = ps.executeQuery();

            // 发送信息到浏览器
            if(rs.next()){
                out.println("该用户已经存在，请换一个名字!");
            }else {
                out.println("OK，这个用户名可用。");
            }
        }catch(Exception e){}
    }
}
```

这就是一个标准的 Servlet 程序，到目前为止 AJAX 只是客户端的 JavaScript 程序。

第一次调试 AJAX 代码的过程让人郁闷，如果得不到想要的效果，先试着在 JavaScript 代码中用 alert 来验证每一步是没问题的，比如，在创建对象后 alert(req);，看看对象是否成功创建，然后到 press 函数中显示一下两个状态。如果 press 中的显示不起作用，那么要考虑 ValiUser 是否有问题，我们可以直接在浏览器的地址栏中输入 req 的请求样式，比如 http://127.0.0.1:8080/360buy/valiUser.bin?user=aaa，看看能不能得到想要的结果。

现在我们再来看 AJAX 的好处，由于只是在需要的时候将一个用户名传到服务器，服务器的程序也只是传送回来了一句话，不但用户响应的速度大幅度地提高了，而且感受也好多了，正确情况下传统的验证会重新装载页面，如果不使用代码进行信息传递，用户曾经输入的信息都会丢失，AJAX 让用户重新找到了单机版程序的响应效果。

3.5.6 用 AJAX 实现分页显示

在实际项目中，商品展示可能只是页面的一个部分，每次选择不同页码后，都要重新刷新网页，用户体验也不好，这时也可以使用 AJAX，只更新表格中的数据。

但是和用户名识别不同的是商品显示的内容比较多，好在 AJAX 这个单词最后的 X 指的是 XML，我们知道 XML 非常适合装载大量的数据，并且很容易被解析，现在在网站上很多复杂数据的传送都是使用 XML 形式。

首先我们要改造商品展示的 Servlet 程序，使其专门提供需要显示的数据，数据使用 XML 来组织，我起了一个新的类名 ProData，我建议你索性从写一遍这个类。具体代码如下。

```
package com.wy;

import javax.servlet.http.*;
import java.io.*;
```

```
import java.sql.*;

public class ProData extends HttpServlet {
   public void doGet(HttpServletRequest request,
         HttpServletResponse response) {
      try {
         // 设置中文显示
         response.setContentType("text/xml;charset=gb2312");

         // 准备输出流
         PrintWriter out = response.getWriter();

         // 准备数据库连接
         Connection cn = DataBase.getConnection();

         //接收当前页码
         int nowPage = 1;
         try{
            nowPage  =  Integer.parseInt(request.getParameter  ("pa
ge"));
         }catch(Exception ee){}
         // 查询数据
         Statement st = cn.createStatement();
         ResultSet rs = st.executeQuery(
   "select img, name, price, gift, down, info, pid from products");

         // 跳过不需要的商品
         for(int i = 0; i < (nowPage-1)*8; i ++){
            rs.next();
         }

         // 输出 XML 基本标记
         out.println("<?xml version=\"1.0\" encoding=\"GB2312\ "?>");
         out.println("<Products>");

         // 查询评论数量
         PreparedStatement ps = cn.prepareStatement(
           "select count(pid) from comment where pid=?");

         for(int i = 0; i < 8; i ++){
            if(rs.next()){
               out.println("<product>");

               out.println("<img>");
               out.println(changCode(rs.getString(1)));
               out.println("</img>");

               out.println("<name>");
```

```
                    out.println(changCode(rs.getString(2)));
                    out.println("</name>");

                    out.println("<info>");
                    out.println(changCode(rs.getString(6)));
                    out.println("</info>");

                    out.println("<price>");
                    out.println(rs.getDouble(3));
                    out.println("</price>");

                    out.println("<commentCount>");
                    ps.setInt(1, rs.getInt(7));
                    ResultSet crs = ps.executeQuery();
                    crs.next();
                    out.println(crs.getInt(1));
                    out.println("</commentCount>");

                    out.println("<gift>");
                    out.println(rs.getInt(4));
                    out.println("</gift>");

                    out.println("<down>");
                    out.println(rs.getString(5));
                    out.println("</down>");

                    out.println("</product>");
                }
            }
            out.println("</Products>");
        } catch (Exception e) {}
    }
    private String changCode(String s) throws UnsupportedEncodingException{
        if(s==null) {
            return "";
        }
        return new String(s.getBytes("ISO-8859-1") , "GB2312");
    }
}
```

XML 的要求比 HTML 严格多了，在 HTML 中大多数的双引号可以用单引号代替，但是 XML 不行，所以有些地方我们要用\"这样的转义符来输出双引号。

写这段代码的过程你是否有这样的感受，相对过去分页显示的代码要容易一些，容易的原因是我们只管输出数据，不需要管显示效果，这就是数据和显示分离的好处。

这个 Java 代码写好后，和过去一样的部署，一样的添加到 web.xml 文件中，先不要写网页上的 AJAX，直接用浏览器访问这个 prodata.bin，如果 XML 有格式的问题，浏览器会显示错误，如果正确，你将看到如图 3-24 所示的效果。

图 3-24

前面的小减号用鼠标单击的话，可以看到有动作，这个元素会折叠起来，这些都说明我们输出的 XML 没问题。

当然最大的可能是没有成功，一旦出错了，头疼的事情就来了，虽然改正错误很容易，但是浏览器并不理你的改变，这时要清除掉浏览器缓存的历史记录，然后在刷新。

现在我们要用 JavaScript 来处理这个 XML 结果，JavaScript 开始的代码一样，创建对象，设置对象，然后是处理程序。这个网页程序还需要使用 Servlet，显然页码和用户名显示都需要 Java 代码，但是为了清楚理解，我们先用一个 HTML 文件尝试一下，这个 HTML 文件将准备好一个空的表格，并且模拟了页码的超链接。具体代码如下。

```
<html>
<script>
function newObject(nowPage){
       if(window.XMLHttpRequest){
           req = new XMLHttpRequest();
       }else if(window.ActiveXObject) {
           req = new ActiveXObject("Microsoft.XMLHttp");
       }

       req.onreadystatechange=press;
       req.open("GET" , "prodata.bin?page="+nowPage , true);
       req.send("");

}
function press() {
       if(req.readyState==4) {
           if(req.status==200) {
               // XML 解析处理程序
}
</script>
<body onload="newObject(1)">
```

```
        <table id="t1" border="1" width="100%">
            <tr align="center">
                <td>a</td>
                <td>a</td>
                <td>a</td>
                <td>a</td>
            </tr>
            <tr align="center">
                <td>a</td>
                <td>a</td>
                <td>a</td>
                <td>a</td>
            </tr>
        </table>
        <a href="javascript:newObject(1)">1</a>
        <a href="javascript:newObject(2)">2</a>
        <a href="javascript:newObject(3)">3</a>
    </body>
</html>
```

这段代码有没有什么不好理解的，为了在测试的时候能够看见表格，我在每个单元格中填入了一个 a，并且设定边框为 1。注意我为表格设定了 id 属性 t1，这是为进一步使用 JavaScript 操作表格做好准备。

现在我们将注意力放到 XML 解析处理程序那个注释的位置。要将内容填到表中，要有能力解析 XML，有能力访问表格，剩下的就是你的逻辑能力了。解析 XML 用 DOM，同时我提供访问表格的代码。

找到第二个商品名称的代码如下，请将这段代码放到注释的位置进行测试。

```
root = req.responseXML.getElementsByTagName("*");
alert(root[0].childNodes[1].childNodes[1].nodeName);
alert(root[0].childNodes[1].childNodes[1].childNodes[0].nodeValue);
```

我用了两个 alert，第一个显示的是节点的名称，第二个显示的是这个元素的值，强烈建议你通过逐步尝试来得到这个代码，或是打印自己假定的其他内容。

比如先看 alert(root[0].nodeName) ;是什么，然后在添加一个 childNodes[1]，看 alert(root[0].childNodes[1].nodeName) ;这样一层层地尝试。

现在看如何操作表格，t1 指的是这个表格，按照 DOM 的写法应该是 document.getELementById("t1") ，但是我们已经知道可以直接简写成 t1 就可以了，用 alert(t1)看看行不行，如果不行，就写全吧。事实上 table 本身就是 XML 格式的，不过，因为 table 的层次相当的固定，为了操作方便，t1 的下一层对象是 rows，访问第二行可以写成 t1.rows[1]，或写成 t1.rows.item(1)，这是一项对象数组技术在 JavaScript 中的应用，再下一层是 cells，可以用同样的写法，比如我想在第二行第二个单元格中写入“Java”，代码如下。

```
t1.rows[1].cells[1].innerHTML="Java";
```

现在能够解析传送过来的 XML 信息，也能够将内容放到表格中，我们可以实现代码了，每一步你都先试试，然后再看我的代码。

表格是 2 行 4 列的，我要确保能够将内容写到每个单元格中，所以我用下面的代码来演练对表格的操作。

```
// XML 解析处理程序
root = req.responseXML.getElementsByTagName("*");
count = root[0].childNodes.length;
for(i = 0; i < count; i ++) {
    row = i/4; // 思考一下我是如何得到行号和单元格位置的
    cell = i%4;
    t1.rows[row].cells[cell].innerHTML = i;
}
```

然后将赋值到每个单元格中的 i，替换成 root[0].childNodes[i].childNodes[1].childNodes[0].nodeValue，这样就能将所有的商品名称填入到相应的表格中了。

在每个单元格中填入的将是复杂的 HTML 代码，所以我用一个变量来准备要填入的 HTML 语句。

全部的 HTML 代码如下。

```
<html>
<script>
function newObject(nowPage){
    if(window.XMLHttpRequest){
        req = new XMLHttpRequest();
    }else if(window.ActiveXObject) {
        req = new ActiveXObject("Microsoft.XMLHttp");
    }

    req.onreadystatechange=press;
    req.open("GET" , "prodata.bin?page="+nowPage , true);
    req.send("");
}

function press() {
    if(req.readyState==4) {
        if(req.status==200) {
            // XML 解析处理程序
            root = req.responseXML.getElementsByTagName("*");
            count = root[0].childNodes.length;
            for(i = 0; i < count; i ++) {
                cell = i%4;
                row = i/4;
                product = root[0].childNodes[i];

                content = "<img src='";
                content += product.childNodes[0].childNodes[0]. nodeValue;
```

```
                content += "'/>";

                content += "<div class='name'>";
                content += "<a href='' title='";
                if(product.childNodes[2].childNodes[0]!=null){
                   content   +=   product.childNodes[2].child   Nodes[0].
nodeValue;
                }
                content += "'>";
                content += product.childNodes[1].childNodes[0]. nodeValue;
                content += "</a></div>";

                content += "<div class='price'>&yen;";
                content += product.childNodes[3].childNodes[0]. nodeValue;
                content += "</div>";

                content += "已有";
                content += product.childNodes[4].childNodes[0]. nodeValue;
                content += "人评价";

                if(product.childNodes[5].childNodes[0].nodeValue!="0"){
                   content  +=  "<a  class='p1'  title='购买本商品送赠品
'><imgsrc='img/gift.png'></a>";
                }

                if(product.childNodes[6].childNodes[0].node Value.charAt
(1)=="y"){
                   content  +=  "<a  class='p2'  title='本商品正在降价销售中
'><imgsrc='img/down.png'></a>";
                }

                content += "<div>";
                content += "<input type='button' value='购买'>";
                content += "<input type='button' value='关注'>";
                content += "<input type='button' value='对比'>";
                content += "</div>";

                t1.rows[row].cells[cell].innerHTML = content;
             }
          }
       }
    }
    </script>
    <link rel=stylesheet href='show.css' type='text/css'>
    <body onload="newObject(1)">
       <table id="t1" width="100%">
          <tr align='center'>
             <td></td>
```

```
            <td></td>
            <td></td>
            <td></td>
        </tr>
        <tr align='center'>
            <td></td>
            <td></td>
            <td></td>
            <td></td>
        </tr>
    </table>
    <a href="javascript:newObject(1)">1</a>
    <a href="javascript:newObject(2)">2</a>
    <a href="javascript:newObject(3)">3</a>

</body>
</html>
```

为了减少代码量，我将 root[0].childNodes[i]保存到变量 product 中，这样就不必每次都写那么长了。

大部分代码都相当的好理解，在降价图片判断的时候，我遇到了问题，因为生成 XML 程序在每个值的上下都进行了换行，这样要用 charAt 取第二个值进行比较才有效。事实上应该修改 ProData 程序，但是为了不重复列举代码，我在 HTML 这一端处理了，其实 info 也有这个问题，都应该修改提供 XML 的后台程序。

虽然我们实现了使用 AJAX 技术分页显示商品，你也能通过这个任务充分理解 AJAX 技术，但是这段 HTML 代码是有严重问题的，我们通过元素的位置来取值，这样一旦后台程序进行了修改，变换了元素位置，页面将出现逻辑问题，所以一定要通过比对 nodeName 来取值，这样程序才会有足够的健壮性，代码修改如下。

```
<html>
<script>
function newObject(nowPage){
    if(window.XMLHttpRequest){
        req = new XMLHttpRequest();
    }else if(window.ActiveXObject) {
        req = new ActiveXObject("Microsoft.XMLHttp");
    }

    req.onreadystatechange=press;
    req.open("GET" , "prodata.bin?page="+nowPage , true);
    req.send("");
}

function press() {
    if(req.readyState==4) {
        if(req.status==200) {
```

```
            // XML 解析处理程序
            root = req.responseXML.getElementsByTagName("*");
            count = root[0].childNodes.length;
            for(i = 0; i < count; i ++) {
                cell = i%4;
                row = i/4;
                product = root[0].childNodes[i];
                nodeCount = product.childNodes.length;

                // 获取 XML 中的信息
                img = "";
                info = "";
                name = "";
                price = "";
                commentCount = "";
                gift = "";
                down = "";
                for(j = 0; j < nodeCount; j ++) {
                    if(product.childNodes[j].nodeName=="img") {
                        img = product.childNodes[j].child Nodes[0].node
Value;
                    }

                    if(product.childNodes[j].nodeName=="info") {
                        if(product.childNodes[j].childNodes[0]!=null){
                            info = product.childNodes [j].childNodes[0].
nodeValue;
                        }
                    }

                    if(product.childNodes[j].nodeName=="name") {
                        name = product.childNodes[j].child Nodes[0].node
Value;
                    }

                    if(product.childNodes[j].nodeName=="price") {
                        price = product.childNodes[j].ch ildNodes[0].node
Value;
                    }

                    if(product.childNodes[j].nodeName=="commentCount") {
                        commentCount = product.childNodes [j].childNodes
[0].nodeValue;
                    }

                    if(product.childNodes[j].nodeName=="gift"){
                        gift = product.childNodes[j].child Nodes[0].node
Value;
```

```
            }

            if(product.childNodes[j].nodeName=="down") {
                down = product.childNodes[j].childNodes [0].nodeValue;
            }
        }

            // 生成单元格中的 HTML 语句
            content = "";
            content = "<img src='";
            content += img;
            content += "'/>";

            content += "<div class='name'>";
            content += "<a href='' title='";
            content += info;
            content += "'>";
            content += name;
            content += "</a></div>";

            content += "<div class='price'>&yen;";
            content += price;
            content += "</div>";

            content += "已有";
            content += commentCount;
            content += "人评价";

            if(gift !="0"){
                content += "<a class='p1' title='购买本商品送赠品'>";
content += "<img src='img/gift.png'></a>";
            }

            if(down=="y"){
                content += "<a class='p2' title='本商品正在降价销售中'>";
content += "<img src='img/down.png'></a>";
            }

            content += "<div>";
            content += "<input type='button' value='购买'>";
            content += "<input type='button' value='关注'>";
            content += "<input type='button' value='对比'>";
            content += "</div>";

            t1.rows[row].cells[cell].innerHTML = content;
      }
   }
   }
```

```
}
</script>
<link rel=stylesheet href='show.css' type='text/css'>
<body onload="newObject(1)">
    <table id="t1" width="100%">
        <tr align='center'>
            <td></td>
            <td></td>
            <td></td>
            <td></td>
        </tr>
        <tr align='center'>
            <td></td>
            <td></td>
            <td></td>
            <td></td>
        </tr>
    </table>
    <a href="javascript:newObject(1)">1</a>
    <a href="javascript:newObject(2)">2</a>
    <a href="javascript:newObject(3)">3</a>

</body>
</html>
```

在写上述代码的过程中，你是否想到，我们可以写得更加简洁，表格可以用 JavaScript 来生成，通过元素的标记名获得其中的值，可以用一个函数来做，我们改造一下代码。

```
<html>
<script>
function newObject(nowPage){
    if(window.XMLHttpRequest){
        req = new XMLHttpRequest();
    }else if(window.ActiveXObject) {
        req = new ActiveXObject("Microsoft.XMLHttp");
    }

    req.onreadystatechange=press;
    req.open("GET","prodata.bin?page="+nowPage, true);
    req.send("");
}

function press() {
    if(req.readyState==4) {
        if(req.status==200) {
            // XML 解析处理程序
            root = req.responseXML.getElementsByTagName("*");
            count = root[0].childNodes.length;
            for(i = 0; i < count; i ++) {
```

```
                cell = i%4;
                row = i/4;
                product = root[0].childNodes[i];

                // 生成单元格中的 HTML 语句
                content = "";
                content = "<img src='";
                content += getValue(product, "img");
                content += "'/>";

                content += "<div class='name'>";
                content += "<a href='' title='";
                content += getValue(product, "info");
                content += "'>";

                content += getValue(product , "name");
                content += "</a></div>";

                content += "<div class='price'>&yen;";
                content += getValue(product , "price");
                content += "</div>";

                content += "已有";
                content += getValue(product, "commentCount");
                content += "人评价";

                if(getValue(product, "gift") !="0"){
                    content += "<a class='p1' title='购买本商品送赠品'>");
content += "<img src='img/gift.png'></a>";
                }

                if(getValue(product , "down") =="y"){
                    content += "<a class='p2' title='本商品正在降价销售中'>");
content += "<img src='img/down.png'></a>";
                }

                content += "<div>";
                content += "<input type='button' value='购买'>";
                content += "<input type='button' value='关注'>";
                content += "<input type='button' value='对比'>";
                content += "</div>";

                t1.rows[row].cells[cell].innerHTML = content;
            }
        }
    }
}
```

```
function getValue(product , name) {
    nodeCount = product.childNodes.length;

    for(j = 0; j < nodeCount; j ++) {
        if(product.childNodes[j].nodeName==name) {
            if(product.childNodes[j].childNodes[0]!=null){
                return product.childNodes[j].childNodes[0].nodeValue;
            }
        }
    }
}
</script>
<link rel=stylesheet href='show.css' type='text/css'>
<body onload="newObject(1)">
    <table id="t1" width="100%">
        <script>
        for(row = 1; row<=2; row ++) {
            document.write("<tr align='center'>");
            for(cell = 1; cell<=4; cell++) {
                document.write("<td></td>");
            }
            document.write("</tr>");
        }
        </script>
    </table>
    <a href="javascript:newObject(1)">1</a>
    <a href="javascript:newObject(2)">2</a>
    <a href="javascript:newObject(3)">3</a>
</body>
</html>
```

我想现在的这个代码已经近乎于完美了，但是我们的工作还没有结束，用户名还无法显示，下面的页码也是假的，我们必须将这个 HTML 文件，转变成 Servlet 才能实现全部的功能。

Servlet 还没开始写，我们就能想到将上面的代码输出的大量语句会很烦人，好在 JavaScript 部分代码可以分离到 js 文件中，然后使用<script src="products.js"></script>，将这个文件引入到 HTML 中，建议在写 Servlet 前，先在 HTML 文件中尝试着这样来做，要注意的是，引入 js 文件的格式不能写成<script src="products.js"/>，必须将标记的头尾分开来写。

现在我们来实现 Servlet。具体代码如下。

```
package com.wy;

import javax.servlet.http.*;
import java.io.*;
import java.sql.*;

public class Products extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse
```

```
response) {
       try {
           // 设置中文显示
           response.setContentType("text/html;charset=gb2312");

           // 准备输出流
           PrintWriter out = response.getWriter();

           // 准备数据库连接
           Connection cn = DataBase.getConnection();
           Statement st = cn.createStatement();

           //接收当前页码
           int nowPage = 1;
           try{
              nowPage = Integer.parseInt(request.getParameter ("page"));
           }catch(Exception ee){}

           // 输出 HTML 基本标记
           out.println("<html>");
           out.println("<link rel=stylesheet href='show.css'type='text
/css'>");
           out.println("<script src='products.js'></script>");
           //显示用户名
           HttpSession session = request.getSession();
           String user = (String)session.getAttribute("user");
           if(user==null){
              out.println("<a href='login.html'>请登录</a>");
           }else {
              out.println("欢迎, "+user);
           }

           out.println("<body onload='newObject(1)'>");
           out.println("<table id='t1' width='100%'>");

           out.println("<script>");
           out.println("for(row = 1; row<=2; row ++) {"};
           out.println("document.write(\"<tr align='center'>\");");
           out.println("for(cell = 1; cell<=4; cell++) {"};
           out.println("document.write('<td></td>');");
           out.println(")");
           out.println("document.write('</tr>');");
           out.println(")");
           out.println("</script>");
           out.println("</table>");
           out.println("");
           out.println("");
           out.println("");
```

```
            //显示页码
            //获得总行数
            ResultSet rs = st.executeQuery("select count(pid) from
products");
            rs.next();
            int rowCount = rs.getInt(1);
            //计算总页数
            int pageCount = 0;
            if(rowCount%8==0){
               pageCount = rowCount/8;
            }else {
               pageCount = rowCount/8 + 1;
            }
            //显示
            for(int i = 1; i <= pageCount; i ++){
               out.println("<a href='show.bin?page="+i+"'>"+i+"</a>");
            }
            out.println("</body>");
            out.println("</html>");
         } catch (Exception e) {}
      }
}
```

提供 XML 数据的 Servlet，js 文件和 CSS 文件我就不提供了，前面的代码都准备了这些内容。在实际的运行环境中，由于网络的延时，得到的效果可能会比你在单机状态下要慢，为此有些人会提前缓存一些内容，使用隐藏的 div 放到浏览器里面，以便得到更好的用户体验。

AJAX 存在一个问题，就是浏览器的后退功能无效了，因为浏览器无法将异步传送过来的内容区分成两个页面，Google 的工程师通过截获用户的后退操作解决了这个问题，如果有兴趣的话你可以自己研究一下。

这样我们使用 Servlet 来实现动态网页的功能就告一段落了，我们实现了用户登录、注册和分页显示商品，你有必要将这些功能联系到一起。

第 4 章

JSP

告诉我编写 Servlet 的感受是什么？这几乎是糟糕透顶的技术，非常烦琐的部署过程，让任何一点代码的修改都伴随着复杂的操作过程，当然很多集成的开发工具解决了这个问题，工具可以自动地添加 web.xml，自动地部署，无需重启 Tomcat。

从编码角度来看，做一个 Java 程序员有多难，我们用 Java 编写 HTML、SQL、JavaScript、CSS，甚至 XML，不搞乱这些不同的语法就已经是高手了。

随着动态网页越来越普遍的使用，人们需要更加简洁的开发技术，技术就是这样进步的，人们发现动态网页就是 HTML+Java 的组合，一直以来我们都是在 Java 基础上写 HTML，毕竟学习 HTML 要比学习 Java 容易得多，能不能在 HTML 上添加 Java 代码，这样有 HTML 基础的人只需要学习部分的 Java 知识就可以上手了，这个思路并不是 Java 最早提出的，在这之前 ASP 先实现了这个思路，Java 照着 ASP 学，而且将其命名为 JSP。

在 Java 基础上写 HTML 用的是输出语句，在 HTML 里面写 Java，Tomcat 如何处理？毕竟 HTML 不是编程语言，从本质上 HTML 无法直接发起计算机的动作，除非有人将这个 HTML+Java 的混合体翻译成 Java 程序，Tomcat 就是这么做的。

我们来看一下 Tomcat 完成翻译的过程是什么样子的，还记得 login.html 那个文件吗？现在将这个文件的扩展名改成.jsp，然后启动 Tomcat，之后用浏览器访问 http://127.0.0.1:8080/ 360buy/login.jsp，稍微停顿了一下，用户登录的页面呈现了出来，但是中文是乱码，先不理这个问题，回头我们再解决。

这就是你的第一个 JSP 程序，相当的简单吧。你是不是有种上了当的感觉。别着急，跟着我的思路向下走。我们一直都是在 webapps 这个目录中写代码，在 webapps 同一级还有一个 work 目录，现在我们到 work 目录中看看。如图 4-1 所示。

这个目录层级非常多，一步步地点进去，竟然在这里也能遇到我们创建的 360buy 目录，如果这个目录是空的，说明你没有成功地访问 JSP 文件，到了最后一层，我们竟然看见了一个 Java 源文件，login_jsp.java，打开这个文件我们看到了一个成员方法_jspService，前面还有加了_jsp 的 init 方法和 destroy 方法，怎么看这都像是 Servlet。

没错这就是 Servlet，问题是这个 Servlet 程序是从哪来的呢？虽然这个程序不是我们写的，但是仔细看方法_jspService 里面，显然有我们在 login.jsp 里面写的 HTML 代码，除了中文在这里已经变成了乱码，无法识别，HTML 的标记就是我们写的，这些内容被 out.write 方法一句句

地进行了输出。

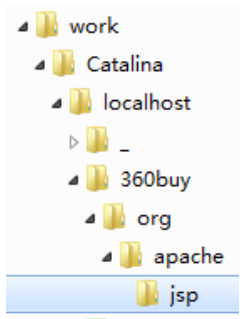


图 4-1

这些证据充分证明这个 Servlet 是从 login.jsp 这个文件来的，Tomcat 在背后做了工作，Tomcat 将我们的 JSP 文件翻译成了 Servlet，并且调用 java_home 这个环境变量提供的 JDK 将 Servlet 编译成了 Class 文件，这些动作是在用户第一次访问 JSP 的时候完成的，所以你访问的时候会感受到一丝停顿，第二次访问就没有这个问题了，除非 JSP 文件发生了改变，否则 Tomcat 会直接使用此前生成好的 Class，也就是说从第二次访问开始，JSP 的运行效率和 Servlet 一样。

在编写并运行 JSP 的过程中，我们没有改写 web.xml，没有部署，没有重启 Tomcat，人做的工作和写 HTML 文件没有区别，但是 Tomcat 做了复杂的事情。

因为你已经有了很好的 Servlet 基础，现在你也知道 JSP 就是 Servlet，并且能够看到生成的 Servlet 代码，那么学习 JSP 对于你来说就非常简单了，遇到 JSP 中新的东西，就用一下，然后去看生成的 Servlet 代码，利用你的 Java 和 Servlet 知识去理解它。

4.1 用户登录

我们还遗留了一个问题要处理，就是用户登录界面中的中文是乱码，如果是 Servlet，我们通过 response.setContentType("text/html;charset=GB2312") ;来设置 HTTP 协议头。

将 Java 代码插入到 HTML 中使用<%//此处放 Java 代码%>，我们现在打开 login.jsp 文件在第一行加入这样的设置，然后重新访问，问题并没有解决，乱码依旧。

去 work 目录打开生成的 Servlet 程序，你发现我们写的这句话赫然放在代码中，问题是在这段代码中有两个 response.setContentType 设置，一个估计是 Tomcat 自动提供的，没有 charset=GB2312，一个是我们写的，显然 Tomcat 在使用这个 Servlet 的时候没用我们的设置。

但也不完全是坏消息，至少我这样写没有报告错误，而且我们写的语句也被成功地添加到了生成的代码中，为什么我会为这件事情感到高兴呢？你发现我直接使用了 response 对象，从 JSP 文件的角度看，让人感到非常的不好理解，整个 JSP 文件中没有对 response 对象的创建，甚至是

说明，传统的解释是 response，也包括 request 是 JSP 的内置对象，不需要创建就自动存在。

但是我们知道是怎么一回事，再看一眼生成的 Servlet 源文件，你会发现在我们的代码前面有很多语句，这些语句大部分都在声明引用变量，既然我们的语句必定会位于这些声明的下面，那么我们就一定可以使用这些声明了的引用变量。

```
public  void  _jspService(final  javax.servlet.http.HttpServletRequest
request,
final javax.servlet.http.HttpServletResponse response)
       throws java.io.IOException, javax.servlet.ServletException {

   final javax.servlet.jsp.PageContext pageContext;
   javax.servlet.http.HttpSession session = null;
   final javax.servlet.ServletContext application;
   final javax.servlet.ServletConfig config;
   javax.servlet.jsp.JspWriter out = null;
   final java.lang.Object page = this;
   javax.servlet.jsp.JspWriter _jspx_out = null;
   javax.servlet.jsp.PageContext _jspx_page_context = null;


   try {
     response.setContentType("text/html");
     pageContext = _jspxFactory.getPageContext(this, request, response,
                     null, true, 8192, true);
     _jspx_page_context = pageContext;
     application = pageContext.getServletContext();
     config = pageContext.getServletConfig();
     session = pageContext.getSession();
     out = pageContext.getOut();
     _jspx_out = out;

 response.setContentType("text/html;charset=GB2312");
     out.write("\r\n");
     out.write("<html>\r\n");
     out.write("\t<body>\r\n");
     out.write("\t\t<form action=\"loginc.bin\" method=\"POST\">\r\n");
     out.write("\t\t\t 用户名<input type=\"text\" name=\"username\"/>
<br/>\r\n");
     out.write("\t\t\t 密码<input type=\"password\" name=\"password\"/>
<br/>\r\n");
     out.write("\t\t\t<input type=\"submit\" value=\"登录\"/>\r\n");
     out.write("\t\t</form>\r\n");
     out.write("\t</body>\r\n");
     out.write("</html>");
   } catch (java.lang.Throwable t) {
     if (!(t instanceof javax.servlet.jsp.SkipPageException)){
       out = _jspx_out;
```

```
        if (out != null && out.getBufferSize() != 0)
          try { out.clearBuffer(); } catch (java.io.IOException e) {}
        if (_jspx_page_context != null) _jspx_page_context.handlePage
Exception(t);
      }
    } finally {
      _jspxFactory.releasePageContext(_jspx_page_context);
    }
  }
```

我将生成的 service 方法的那个部分代码列举出来，并且将声明的变量加黑了，这些变量就是所谓的 JSP 内置对象，数一数一共有 8 个，而 JSP 的内置对象一共有 9 个，多出来的那个是 exception。遵循我的传统，我不会一个个地解释每个所谓的内置对象是什么，用到了我们再说，至少到目前为止 request 和 response 不用我来解释，就是 Servlet 里面对应的对象。

4.1.1 设置中文编码

我们的问题还是没有解决，中文乱码该如何解决，显然在<%%>中编写代码不行，因为这里的代码影响不到整个 Servlet 的设置，<%%>中运行的都是 Java 语句，在 JSP 中称其为“脚本元素”，JSP 另外提供了<%@%>称作“指令元素”，我们可以理解为<%%>中的脚本是战术层面的语句，而<%@%>中的指令是战略层面的语句，无法写入到方法中的语句，都有理由尝试着到指令元素中找有没有适合的语句。

在我们的程序中，设置中文的代码是 <%@ page contentType="text/html;charset=GB2312" %>，第一个单词是 page，所以我们进一步称其为“page 指令”，这样说来就会有其他指令，page 指令也有很多，我们现在用的是 contentType，page 指令的代码会影响这个页面，也就是影响整 个 Servlet，再次访问就会看到中文了，然后你去 work 中看看生成的代码，有什么地方发生了改变。

现在我们要编写登录的处理程序 loginc.jsp，在 360buy 目录中创建一个新的文本文件 loginc.jsp，然后检查一下 login.jsp 中表单的 action 属性，确保提交表单的处理程序是 loginc.jsp，现在在 loginc.jsp 中写入如下代码。

```
<%@page contentType="text/html;charset=GB2312"%>
```

用户登录处理程序

```
<%int i = 1+1 ;%>
```

处理结束

4.1.2 编写脚本

保存后用浏览器访问，不需要重启 Tomcat。你能看见打印出来的文字，查看生成的 Servlet，可以看到脚本元素的 Java 语句是按照文档流的顺序安排在生成的代码中的。request 对象完全不需要我介绍，你也应该会使用，下面的任务你自己尝试着完成，接收用户名和密码，

如果用户名是“aaa”，密码是“111”，就显示“欢迎”，否则显示“登录失败”。

```
<%@page contentType="text/html;charset=GB2312"%>
<%
    String user = request.getParameter("username");
    String pass = request.getParameter("password");
    if(user.equals("aaa")&&pass.equals("111")) {
        out.println("欢迎");
    }else {
        out.println("登录失败");
    }
%>
```

这个 JSP 里面是彻头彻尾的 Java 语句，需要说明的是，输出结果的时候，你看到使用了 out 对象，不用说，这也是内置对象，也能在生成代码的方法中找到 out 的声明，其实在本书 Servlet 部分中我已经在使用 out 作为指向浏览器的输出流了，为的就是和这个 out 统一，只是那时候的 out 是 PrintWriter 对象，而这里是 JspWriter 对象，这两个类非常相似，但是并不相同，之间的细微区别，如果你有兴趣的话可以研究一番。

要说以 loginc.jsp 开始学习 JSP 有点不恰当，我们知道 JSP 就是向 HTML 中加入 Java 语句，但是这个程序几乎就没有 HTML，也不是，“欢迎”和“登录失败”就是 HTML，只是我们写到 Java 里面去了，所以我们在这个程序中可以将 HTML 代码分离出去，要注意的是未来生成 Servlet 时，Java 的脚本代码和 HTML 之间会遵循原有的顺序。

```
<%@page contentType="text/html;charset=GB2312"%>
<%
    String user = request.getParameter("username");
    String pass = request.getParameter("password");
    if(user.equals("aaa")&&pass.equals("111")) {%>
        欢迎
    <%}else {%>
        登录失败
    <%}
 %>
```

建议换一句输出的话来测试，这样能够确定是新的代码起了作用。如果你能够理解上面的代码，我们再进行进一步的学习，因为这种一会儿 Java，一会儿 HTML 容易让人出错，但是你必须适应这样的写法，这就是 JSP。

无论写了什么，都不要忘记去看一眼生成的 Servlet，这有助于你深入理解 JSP 做了什么。

4.1.3　连接数据库

下一步应该实现到数据库中进行验证，还是你先写。

在写连接数据库的代码时，我们遇到了困难，没法导入 java.sql.*，如果写到<%%>中，导入这句话就会写在方法中，我们知道导入语句只能写在类的外面开始的位置。如果真是不知道如

何导入（import），倒也有办法，就是使用类的时候，带着包名写。具体代码如下。

```
<%@page contentType="text/html;charset=GB2312"%>
<%
    String user = request.getParameter("username");
    String pass = request.getParameter("password");
    Class.forName("org.gjt.mm.mysql.Driver");
    java.sql.Connection cn = java.sql.DriverManager.getConnection
        ("jdbc:mysql://127.0.0.1:3306/360buy","root","123456");
    java.sql.PreparedStatement ps = cn.prepareStatement
        ("select * from user where username=? and password=?");
    ps.setString(1, user);
    ps.setString(2, pass);
    java.sql.ResultSet rs = ps.executeQuery();
    if(rs.next()) {
       response.sendRedirect("show.jsp");
    }else {
       response.sendRedirect("reg.jsp");
    }
 %>
```

现在程序比较小还好办，程序大了总是连着包写类确实麻烦，还是要找到导入包的办法，其实在前面的内容中，我已经给出了提示，这种在<%%>中解决不了的问题，要看看能不能在<%@%>中找到解决办法。有一个 page 指令<%@page import="java.sql.*"%>，要注意这是一条 JSP 指令，不是 Java 的语句，所以没有分号，如果要导入两个包，可以用逗号来分隔，如<%@page import="java.sql.*,java.io.*" %>。一旦导入了 java.sql.*，我们就可以直接使用类，不需要连着包名一起写。

4.1.4 跳转

通常不会就显示一句话，之前我们已经讨论过这个话题，相应的位置应该是跳转到另外的网页中去。几乎不需要我告诉你什么，你就应该能实现这样的跳转。

```
<%@page contentType="text/html;charset=GB2312"%>
<%
    String user = request.getParameter("username");
    String pass = request.getParameter("password");
    if(user.equals("aaa")&&pass.equals("111")) {
       response.sendRedirect("show.jsp");
    }else {
       response.sendRedirect("reg.jsp");
    }
 %>
```

和 Servlet 的语句没有区别。因为 show.jsp 和 reg.jsp 还没有做，所以跳转后会报告 404 错误，找不到文件，但是可以通过这个错误看到跳转成功了。

但是我要告诉你另外的跳转指令<jsp:forward page="show.jsp"/>，使用这个跳转指令来修改

代码看看。

```
<%@page contentType="text/html;charset=GB2312"%>
<%@page import="java.sql.*,java.io.*" %>
<%
    String user = request.getParameter("username");
    String pass = request.getParameter("password");
    Class.forName("org.gjt.mm.mysql.Driver");
    Connection cn = DriverManager.getConnection
         ("jdbc:mysql://127.0.0.1:3306/360buy","root","123456");
    PreparedStatement ps = cn.prepareStatement
         ("select * from user where username=? and password=?");
    ps.setString(1, user);
    ps.setString(2, pass);
    java.sql.ResultSet rs = ps.executeQuery();
    if(rs.next()) {%>
       <jsp:forward page="show.jsp"/>
    <%}else {%>
       <jsp:forward page="reg.jsp"/>
    <%}
 %>
```

这个代码也可以成功地实现跳转，为什么 JSP 要提供两种跳转的形式呢？这个问题你先思考并尝试着找到答案，你可以比较使用两种语句时页面上不同的表现，也可以查看生成的 Servlet 代码有何不同，站到设计 JSP 那个人的角度思考提供两个跳转的必要性。

第一种跳转语句其实不用看生成的代码，那就是标准的 Java 语句，一定是原封不动地写到了方法里面，第二种跳转方式产生的代码如下。

```
if (true) {
   _jspx_page_context.forward("show.jsp");
    return;
}
```

前面的方法没见过，但是可以想象得到，应该就是能产生跳转到 show.jsp 效果的语句，后面跟着的 return 作用太清楚了，先抛开 response.sendRedirect("show.jsp");和 _jspx_page_context.forward("show.jsp");的不同不说，这里多个 return，是不是就意味着使用我们之前使用的跳转语句后面没有 return，如果后面还有代码，是不是后面的代码还要运行？

而且这个 if 判断也很怪异，有点画蛇添足的感觉，事实上这个 if 语句是有用处的，后面我们会讨论这个 if 语句。

看来我要写代码来验证一下我的想法，我在两个不同跳转指令的后面显示窗体，当然现实的项目中不大可能在 JSP 中出现这样的代码。

```
<!-- 这是 jsp1.jsp -->
<% response.sendRedirect("login.jsp");
java.awt.Frame f = new java.awt.Frame();
f.setSize(300 , 400);
f.show(true);
```

```
%>
```

用浏览器访问这个 JSP 文件，我们发现窗体显示了出来。有意思的是，因为 awt 的窗体自身没有关闭的能力，你知道在什么地方能关闭这个窗体吗？我们能看见这个窗体是因为现在服务器和浏览器都在我们这一台计算机上，如果是正常状态，服务器和浏览器分布在两台不同的计算机上，你觉得窗体是显示在浏览器这台计算机上，还是显示在 Tomcat 服务器的计算机上呢？这个窗体是和 Tomcat 在一台计算机上出现的，因此要关闭 Tomcat 才能关闭这个窗体，所以我之前说这样的一条语句对于客户来说是没有意义的，因为客户看不见。

```
<!-- 这是 jsp2.jsp -->
<jsp:forward page="login.jsp"/>
<% java.awt.Frame f = new java.awt.Frame();
f.setSize(300,400);
f.show(true);
%>
```

用浏览器访问上面这个 JSP 文件，窗体没有显示。

这样至少可以得出一个结论，使用 response.sendRedirect 跳转，之后的代码也会运行。而用<jsp:forward />跳转，后续的代码将失去作用，但是这个结果并不能让我们弄明白为什么会有两个跳转。

在 jsp1.jsp 的跳转指令后面加上 return;，岂不是和第二个 JSP 的效果一样，你可以试一下，这样做是不允许的，Java 相当的聪明，它能够识别出来，如果 return 了，那么后面的代码将不可达到，编译的时候就会直接报错，类似的情况是如果你在死循环的后面写语句，也会报这样的错误。但是 Java 的聪明也仅仅到这个程度，我们还是可以欺骗它一下。

```
<!-- 这是 jsp1.jsp -->
<% response.sendRedirect("login.jsp");
if(true) {
  return;
}
java.awt.Frame f = new java.awt.Frame();
f.setSize(300,400);
f.show(true);
%>
```

这样做就不会报错，回想一下用<jsp:forward/>生成的代码，里面是不是有这个看起来有些无聊的 if 语句呢？

一定还有其他什么地方不同。你再运行一遍 jsp1.jsp，在看到用户登录界面的时候，你看一下浏览器的地址栏上是 jsp1.jsp 还是 login.jsp。然后运行一遍 jsp2.jsp，看到用户登录界面后，你再看一下浏览器的地址栏上是 jsp2.jsp 还是 login.jsp。你能试着解释一下它们两者在这个角度的不同吗？

response.sendRedirect 的跳转机制如图 4-2 所示。

浏览器请求的是 jsp1.jsp，Tomcat 便找到这个 JSP，运行后将网页结果发送给了浏览器，浏览器在解释这个网页时发现了需要跳转，于是发出新的对 login.jsp 的请求，Tomcat 重新找到 login.jsp，

运行后将网页结果再次发送到浏览器。也就是说这个跳转过程浏览器请求了两次，服务器也发送了两个网页。

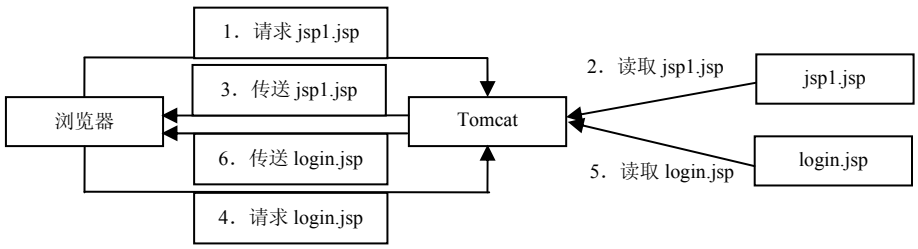


图 4-2

再来看<jsp:forward page="login.jsp"/>是如何跳转的，如图 4-3 所示。

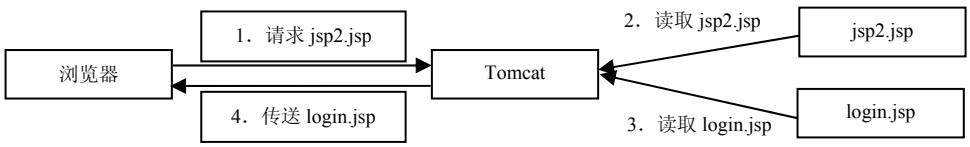


图 4-3

浏览器请求 jsp2.jsp，Tomcat 找到 jsp2.jsp，运行的过程中发现要跳转到 login.jsp，于是直接找到 login.jsp，运行后将网页结果发送到浏览器。

所以使用<jsp:forward/>的跳转，地址栏上依然是 jsp2.jsp，而使用 response.sendRedirect 跳转，地址栏变成了 login.jsp，因为浏览器又对 login.jsp 进行了请求。

很明显<jsp:forward/>运行效率更高，对网络的依赖更少，事实上还有很多以<jsp:开头的元素，我们将这些叫做动作元素。

但是为什么要保留脚本元素的跳转呢？这是因为相比动作元素的跳转，脚本元素可以向下个页面传递参数，还记得分页的代码，我们伪装了表单传递的？page=5 吗？这样第二个页面就能得到参数，很多时候我们不得不传递类似的参数，这样就不得不用脚本元素的跳转。

4.2　购物网站的商品展示

通过用户登录的例子，我们能够理解 JSP 能够完成的工作和 Servlet 是一样的，只是它们编写的方式不同，我们学习这两个技术的重点在于感受到它们之间的不同，所以和学习 Java 不同，我并不会不断地提供新的任务，JSP 阶段主要是重复 Servlet 的任务，只是偶尔会添加一些功能，但是由于 JSP 编写起来相对容易，所以建议重新完成这些任务，并做得更好一些。

做得更好不仅仅是页面做得更漂亮，功能更丰富，作为一个程序员，很重要的一点是编写出高质量的代码，使代码的可读性、健壮性和扩展性更好。我们思考 Servlet 实现该功能的过程，将所有做过的事情划分成几个大的部分，在这个过程中，你可以思考这样做的好处是什么。

```
<html>
<body>
<!-- 准备 -->
<!-- 获取信息 -->
<!-- 显示用户名 -->
<!-- 跳过之前的记录 -->
<!-- 显示当前记录 -->
<!-- 显示页码 -->
</body>
</html>
```

中间的注释就是我的划分，现在我一部分一部分地提供代码，你完全可以利用之前 Servlet 的逻辑和 JSP 知识，逐项、独自地完成这个任务。

```
<!-- 准备 -->
<%@ page contentType="text/html;charset=gb2312" %>
<%@ page import="java.sql.*" %>
<%
    Class.forName("org.gjt.mm.mysql.Driver");
    Connection cn = DriverManager.getConnection
("jdbc:mysql://127.0.0.1:3306/360buy","root","123456");
 %>
```

在准备阶段，我们主要是设置中文输出，准备好数据库连接，这样我使用了两个 page 指令元素，这两个 page 指令我们都接触过，其他的 page 指令你自行到网上搜索一下，看看每一个的作用，这里要说明的是，在所有的 page 指令中，除了 import 外，其他的指令在一个网页中不可以重复，这个要求很好理解，你不能在一个网页的一处设置了 charset=GB2312，又在另一处设置 charset=ISO-8859-1，这样 Tomcat 就不知道该怎么做了，你会觉得自己不可能犯这样的错误，其实在一种类型的应用中存在着这样的风险，到时候我再说。

import 指令是可以重复出现的，但是人们更倾向于使用逗号分隔调入的不同的 java 包。

在获取信息的部分，我们需要的信息有数据结果集 ResultSet、当前页码、总页数、每行显示商品数量，每页显示行数。这里准备的信息数量比用 Servlet 时要多，有些信息过去只是一个数字，现在我们用变量来提供，这样可以在未来进一步提供新功能时减少代码的改变，比如，我们设置每页显示的行数和每行显示的商品数时，因为是用变量提供的，所以未来可以在网页上提供每页显示商品数量的选项，接受用户选择后只需修改变量值，便可提供这样的功能。

```
<!-- 获取信息 -->
<%
    // 每行显示商品数量
    int cells = 4;

    // 每页显示行数
    int rows = 2;

    // 获得商品数量
    Statement st = cn.createStatement();
```

```
    ResultSet rs = st.executeQuery("select count(pid) from products");
    rs.next();
    int count = rs.getInt(1);

    // 计算总页数
    int pageCount = 0;
    if(count%(cells*rows)==0) {
        pageCount = count/(cells*rows);
    } else {
        pageCount = count/(cells*rows)+1;
    }

    // 获取当前页码
    int nowPage = 1;
    try {
        nowPage = Integer.parseInt(request.getParameter("page"));
    }
    catch (Exception ex) {}

    // 获取商品信息结果集
    rs = st.executeQuery("select img , name , price , gift , down ,
info , pid from products");
 %>
```

显示用户名之前要修改 loginc.jsp，在用户验证成功的时候，将用户名存入 session 中，我不通过 loginc.jsp 了，你应该有能力完成这样的工作。下面是 products.jsp 显示用户名的代码。

```
<!-- 显示用户名 -->
<% String user = (String)session.getAttribute("user");
    if(user==null){%>
        <a href="login.jsp">请登录</a>
    <%}else {%>
        欢迎，<%=user%>
    <%}
%>
```

在两个地方需要注意，session 可以直接使用，不需要从 request 对象中获得，它是 JSP 已经声明好的内置对象。

<%= %>是简洁的打印表达式，效果类似于<%out.println() ;%>，通常用于将一个变量或是一个简单的运算结果直接显示到网页上。

跳过之前的记录相对简单，和 Servlet 的代码没有不同，唯一的一点区别是在 Servlet 中，每页显示几个商品是写成固定的 8 的，而在这里要用行数和每行的商品数相乘运算得来。

```
<!-- 跳过之前的记录 -->
<%
    for(int i = 0; i < (nowPage-1)*(rows*cells); i ++) {
        rs.next();
    }
```

```
%>
```

下面是最复杂的显示商品的代码，我没写完，因为写到这遇到了问题。

```
<!-- 显示当前记录 -->
<table border="1" width="100%">
<%
    for(int i = 0; i < rows; i ++) {%>
        <tr align="center">
        <% for(int j = 0; j < cells; j ++) {
            if(rs.next()){
            %>
            <td>
            <!-- 图片 -->
            <img src="<%=rs.getString(1)%>"/>
            <!-- 商品名称 -->
            <div id="name">
                <a href="" title="<%=rs.getString(6)%>"><%= rs.
getString(2)%></a>
            </div>
            </td>
        <%
            }
        }%>
        </tr>
    <%}
 %>
</table>
```

使用 Java 的循环生成<tr>和<td>来提供表格，这些没有问题，要注意的是，<%%>里面使用的是 Java 语法，所以循环增量 i 和 j 需要用 int 来声明，我们现在的代码中还没有使用 JavaScript，一旦用上了 JavaScript，就很容易出错，JavaScript 的循环不用 int 来声明变量，除非你非常清晰，知道现在写的是 Java 还是 JavaScript，否则写错语法是常见的事情。

输出字段内容使用<%=%>表达式并不难理解，但是你发现没有，我们的代码写在双引号中，所以有些书说<%%>的语句优先执行，执行完成后，再与 HTML 语句合并，这只是一种让人容易理解的说法，事实上是生成 Servlet 源代码的过程中，Tomcat 协调了 HTML 和 Java 语句的顺序。

现在的代码运行是能够看到图片和一些商品名字信息的，但我再次遇到中文问题，因为我们已经设置了网页上的字符集为 GB2312，所以问题不在网页代码输出上，问题是从数据库取值时，编码不一致，过去在 Servlet 中，我们使用了一个转换字符集的方法，这个方法在很多地方需要使用，我也希望在 JSP 文件中也写一个转换字符集的方法，这时遇到了问题。

我们知道写在<%%>中的语句，将被放置在_jspService 方法中，现在我们要在_jspService 方法平级的位置写新的方法，而不是放到这个方法的里面。JSP 提供了一个声明全局变量或方法，或者叫做声明类成员变量或方法的符号<%! %>。

在我们的任务中，声明类成员方法的代码如下。

```
<!-- 声明成员方法 -->
<%!
private String changCode(String s){
    String code = "";
    try {
        if(s!=null) {
            code = new String(s.getBytes("ISO-8859-1") , "GB2312");
        }
    }
    catch (Exception ex) {}
    return code;
}
%>
```

用浏览器访问一下这个 JSP，然后看看生成的代码是如何处理<%! %>的。

继续将显示部分代码补全。

```
<!-- 显示当前记录 -->
<table width="100%">
<% PreparedStatement ps = cn.prepareStatement
("select count(pid) from comment where pid=?");
    for(int i = 0; i < rows; i ++) {%>
        <tr align="center">
        <% for(int j = 0; j < cells; j ++) {
            if(rs.next()){%>
            <td>
            <!-- 图片 -->
            <img src="<%=rs.getString(1)%>"/>

            <!-- 商品名称 -->
            <div id="name">
                <a href="" title="<%=changCode(rs.getString(6))%>">
                <%= changCode(rs.getString(2)) %></a>
            </div>

            <!-- 显示单价 -->
            <div class='price'>&yen;<%= rs.getString(3) %></div>

            <!-- 显示评论 -->
            <% ps.setInt(1, rs.getInt(7));
            ResultSet crs = ps.executeQuery();
            crs.next();%>
            已有<%= crs.getString(1) %>人评论

            <!-- 显示赠品图标 -->
            <%    if(rs.getInt(4) !=0){%>
                <a class="p1" title='购买本商品送赠品'><imgsrc='img/gift.
```

```
png'></a>
            <%} %>

            <!-- 显示直降图标 -->
            <%    if(rs.getString(5).equals("y")){%>
                   <a class="p2" title='本商品正在降价销售中'>
<img src='img/down.png'></a>
            <%}%>

            <!-- 显示按钮 -->
            <div>
            <input type="button" value="购买">
            <input type="button" value="关注">
            <input type="button" value="对比">
            </div>
            </td>
        <%}
        }%>
        </tr>
    <%}
 %>
</table>
```

过去的 js 文件和 CSS 文件还可以用，但是希望你在之前的基础上，能做得更漂亮一些。

以下是显示页码的代码。

```
<!-- 显示页码 -->
<% for(int i = 1; i <= pageCount; i ++) {%>
    <a href="products.jsp?page=<%= i%>"><%= i %></a>
<%} %>
```

4.3 将用户登录结合到商品展示页面中

在京东网站的最上面一直都有一个灰色的条，这个很像是我们在学习 HTML 部分的搜狐首页的菜单条，在很多网站上你都能看见这样的顶部菜单条，而且这个菜单条通常会出现在每一页中。如图 4-4 所示。

图 4-4

现在我们来实现这个菜单条，并且让它出现在每一页中。我在下面的代码中暂时准备了一个样子，除了登录和注册有超链接外，其他的部分都只是显示出来而已。

```
<%@ page contentType="text/html;charset=gb2312" %>
<html>
    <style type="text/css">
```

```
#title{
    background-image:url("img/20111221C.png");
    position:absolute;
    top:0;
    left:0;
    width:103%;
    height:25px;
    border:1px solid lightblue;
}
#item{
    position:absolute;
    left:300px;

}
ul{
    width:800px;
    font-size:12px;
}
li{
    list-style-type:none;
    float:left;
    margin:4px 10px;
}
#plac{
    border-left:1px solid CCCCCC;
}
</style>
<body>
    <div id="title">
        <div id="item">
            <ul>
                <li>您好！欢迎来到京东商城！</li>
                <li><a href="login.jsp">[登录]</a></li>
                <li><a href="reg.jsp">[免费注册]</a></li>
            <li id="plac"></li>
            <li>我的订单</li>
            <li id="plac"></li>
            <li>特色栏目</li>
            <li id="plac"></li>
            <li>移动京东</li>
            <li id="plac"></li>
            <li>企业服务</li>
            <li id="plac"></li>
            <li>客户服务</li>
            </ul>
        </div>
    </div>
</body>
```

```
</html>
```

我们要添加一个功能，如果用户此前已经登录了，那么就将“您好！欢迎来到京东商城!”换成“你好,”+用户名+“!”，我只列出局部的代码。

```
<%String user = (String)session.getAttribute("user");
if(user==null){ %>
    <li>您好！欢迎来到京东商城！</li>
<% }else{ %>
    <li>你好，<%= user %>!</li>
<% } %>
```

测试的时候，要登录，然后手工地输入对 menu.jsp 的访问，有可能还需要对 menu.jsp 刷新一下。

4.3.1 使用 Cookie

不过我们在使用京东商城的时候发现，有时打开计算机后第一次访问这个网站时，你的名字已经出现在菜单条上了，根本不需要登录。这个名字从哪来的呢，不可能是 session，因为关闭浏览器，再次打开对于 Tomcat 来说是两个 session，除非 Tomcat 能够识别出是那台计算机，而不是识别那个浏览器，识别计算机对 Tomcat 来说要求太高了，设计这个体系的人有更简单的办法，将一些信息存放到浏览器所在的计算机里，存放在计算机的硬盘中，这样即便是重启计算机，这些信息还在，这样的技术叫做 Cookie，我们先来看一下如何使用 Cookie 将用户名信息存放到客户端的硬盘上。

可以理解，写 Cookie 的代码是写在 loginc.jsp 中，当用户登录成功后，我们不但要将用户名存入 session，同时要将用户名通过 Cookie 存入客户端计算机的硬盘上。

```
<%@page contentType="text/html;charset=GB2312"%>
<%@page import="java.sql.*,java.io.*" %>
<%
    String user = request.getParameter("username");
    String pass = request.getParameter("password");
    Class.forName("org.gjt.mm.mysql.Driver");
    Connection cn = DriverManager.getConnection
("jdbc:mysql://127.0.0.1:3306/360buy","root","123456");
    PreparedStatement ps = cn.prepareStatement
("select * from user where username=? and password=?");
    ps.setString(1, user);
    ps.setString(2, pass);
    java.sql.ResultSet rs = ps.executeQuery();
    if(rs.next()) {
        session.setAttribute("user", user);
        Cookie username = new Cookie("user" , user);
        response.addCookie(username);
        %>
        <jsp:forward page="products.jsp"/>
```

```
<%}else {%>
    <jsp:forward page="reg.jsp"/>
<%} %>
```

但是上述代码并没有真正将信息存入硬盘中，原因是这个 Cookie 没有那么长的寿命，在不设置它的生命周期的情况下，默认 Cookie 在用户浏览器管理的内存中存活，也就是说，如果用户关闭的浏览器，那么内存中的 Cookie 也就随之消失了，如果想要存入硬盘，我们需要设置 Cookie 的生命周期。

```
Cookie username = new Cookie("user" , user);
username.setMaxAge(60*60*24*365);
response.addCookie(username);
```

因为生命周期的时间单位是秒，所以我们如果设置 1 年的生命周期，就要 60 秒成为 1 分钟，60 分钟成为 1 小时，24 小时成为 1 天，365 天成为 1 年。

当这段代码运行后，Cookie 就存入到用户计算机的硬盘中了，那么我现在好奇它是怎么存在硬盘中的，毫无疑问是以文件的形式存放的，现在我们来找一下这个文件。

通常在浏览器的“Internet 选项”中能够找到“浏览历史记录”中能够找到 Cookie 所在的目录，你也可以试着找 C:\Users\aaa\AppData\Roaming\Microsoft\Windows\cookies 这个目录。找到 Cookie 不容易，并不是一定能找到，找不到可能的原因有两个。

1．更新的操作系统，在 windows 早期的操作系统中，一定能够找到 Cookie 文件，而且文件的名字一眼就能看见，但是由于之前的这几年，Cookie 的存放方案发生了改变，所以 Cookie 文件更加隐蔽。为什么要进行改变？这个我回头再说，但是记住，改变一定是有原因的。

2．用户在浏览器的“Internet 选项”中设置了“隐私”选项卡，阻止了写入 Cookie。

现在我先来解释为什么 Cookie 的设置和隐私有关，在我们这个例子中，Cookie 存放的是用户名，在用户访问这个网站的时候，网站代码会读客户端的 Cookie 信息，如果发现有这个用户名，那么会进行自动登录，而 Cookie 文件是一个 txt 的文本文件，在我们的代码里，Cookie 对象中的 user 字符串和对应的值都以明码的形式存放在文件中，当用户的电脑被别人使用，或用户访问网站的时候，使用的就是网吧等场所的公共计算机，那么用户名等敏感信息就可能被别人看到，这样就会发生隐私的泄露，所以浏览器提供了“隐私”设置，允许用户设定对 Cookie 的阻止。这样也解释了，为什么新的操作系统设置了重重障碍，避免用户轻而易举的找到并打开 Cookie 文件。

我们了解了一般情况下 Cookie 对隐私的威胁，似乎感觉只要自己的计算机不被别人碰到就不用担心这个问题，但是一些专业的黑客会有更高级的手段来窃取 Cookie 的值。我举个例子，假如我们开发了一个留言板，那么用户在留言板上输入的东西就会显示在网页上，这个过程可能是输入的内容在提交到服务器后，被存入数据库的表中，然后程序读数据库的内容生成网页，这个平淡无奇的过程看不出有什么问题，但是如果黑客在输入框中输入的是 HTML，那么这些 HTML 代码再次被整理成网页时，就会被浏览器解析，如果输入的 HTML 代码中包含了 javaScript，javaScript 是有能力读取 Cookie 的，那么这个 javaScript 的代码就可以将读到的东西

用地址栏重写的方式提交到另外的一个 IP 地址的 Tomcat 去，然后黑客在那个地方使用 Servlet 或 JSP 来读取这些信息。

为此，很多论坛、留言板这些允许用户输入的地方，使用代码逻辑检测并阻止用户输入 javaScript 甚至 HTML，Cookie 存入的信息往往也进行加密存放。但是这也是道高一尺魔高一丈的事情，聪明的黑客往往会想到其他系统的弱点，通常只有遭受了攻击后，我们才知道要去防范，有经验的程序员和没有经验的区别往往就在这些上面，而不是简单的技术点。

Cookie 的使用范围可能比我们想象的还要广泛，在没有 Cookie 的情况下，Tomcat 是无法区分用户的，所以所有的用户访问一个网站的时候都将看到完全一样的网页，我们称这样的网站是无模式的网站。有了 Cookie 后，如果一个人曾经访问过这个网站，Tomcat 就有可能通过读取 Cookie 了解到是谁来访问，就可以给这个人提供个性化的页面。比如网站可能会通过 CSS 提供几个样式，用户可以选择自己喜欢的样式，以后每次访问的时候都会自动展现出用户选择的样式；又比如一个用户在购物网站访问的过程，会被程序分析出来，在用户再次访问的时候，首页将推荐用户可能感兴趣的商品，一个没有孩子的用户，在明显的位置上不会看到母婴产品的宣传。我们还可能做到，用户上次在网站上浏览时，关闭了浏览器，再次登录这个网站时，自动提供上次关闭时正在访问的页面。

你可以思考一下，我所举的这些 Cookie 应用的例子，如果用程序实现该怎么做，需要保存一些什么样的信息到 Cookie 中。

现在我们来看读取 Cookie 信息的代码。

```
<% Cookie cs[] = request.getCookies();
String user = "";
for(Cookie c: cs){
    if(c.getName().equals("user")){
        user = c.getValue();
    }
}

if(!user.equals("")){%>
    <li>你好，<%= user %>!</li>
<%} else{
    user = (String)session.getAttribute("user");
    if(user==null){ %>
        <li>您好！欢迎来到京东商城！</li>
    <%    }else{ %>
        <li>你好，<%= user %>!</li>
    <% }
} %>
```

Cookie 是一个对象一个对象添加到客户端的，但是读取的时候，只有一种方法，就是一次性地将所有的 Cookie 都读过来，这是从性能角度考虑的，为了降低网络的使用，读来后通过 Cookie 的名字来检索对应的值。

需要说明的是在 Servlet 中，也完全可以使用 Cookie。

4.3.2　将两个网页合并

menu.jsp 编写到现在就已经够复杂的了，但是我们也只写了最简单的登录和注册的超链接，要知道后面还有几个是有下拉菜单的内容，那个要复杂得多，在京东商城的网站上，每个页面都有这样一个菜单条，总不能每个页面都写一遍 menu.jsp 的代码吧。

为了解决这个问题，JSP 提供了网页合并的代码。

```
<%@ include file="menu.jsp" %>和<jsp:include page="menu.jsp"/>
```

你可以分别将这两条语句放到 products.jsp 适合的位置，发现都能让你高兴地看到，无论那一种形式都能完成任务，这怎么又是两个功能相同的语句。按照惯例，我们来探索它们的不同。为了方便探索，我们在 jsp1.jsp 中写入内容 aaaaaaaaaa。在 jsp2.jsp 文件中写入内容 bbbb<%@ include file="jsp1.jsp" %>bbbbbbb。

先别急着运行，我们到 work 目录，生成 java 文件的子目录中，将所有生成的文件都删除掉，目的是为了能够清楚生成了什么。然后再去浏览器访问 jsp2.jsp，因为 include 代码在 jsp2.jsp 中，所以不能访问 jsp1.jsp，我们能够看到一串 a 出现在一串 b 的中间，这是我们预想到的结果，但是关键在于，我们要到 work 中看看产生了什么，结果发现只产生了一个 java 文件 jsp2_jsp.java，以及它对应的 jsp2_jsp.class。

现在我们将 jsp2.jsp 中的代码改成 bbbb<jsp:include page="jsp1.jsp"/>bbbbbbb。同样清除掉 work 目录中的文件，然后再来访问，结果发现生成了两个 java 文件和两个对应的 class 文件。

我想你该知道这两个 include 的不同了吧。如图 4-5、图 4-6 所示，是两个语句的运行过程，即<%@ include file="jsp1.jsp" %>的运行过程和<jsp:include page="jsp1.jsp"/>的运行过程。

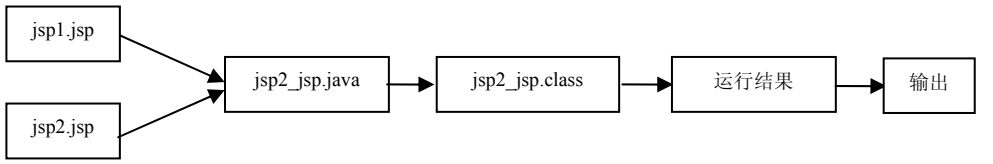


图 4-5

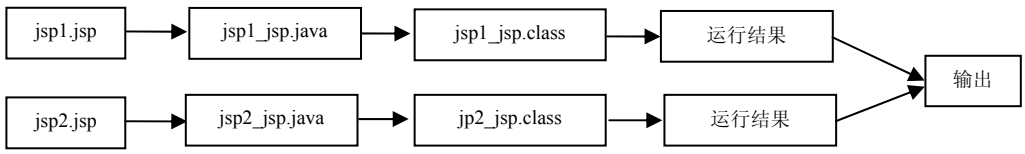


图 4-6

我们将这两种网页合并的方式简单地总结成，先合并和后合并。那么，有这样两个选择，我们该如何选呢？我们以目前做的京东商城的例子来看，假设 menu.jsp 最终生成的 class 文件的大

小是 20KB，使用先合并的方式，如果将 20 个页面合并，就会占用 400KB 的服务器内存，后合并的方案只占 20KB，当然在多用户并发访问的情况下，实际的内存占用远超过这个数量。但是后合并的方案在运行过程中，因为多出来合并的过程，所以运行起来会多消耗一点时间。也就是说先合并浪费内存，但是速度快，而后合并节约内存，但是速度慢，看你的选择策略了。

我想在这个基础上，你可以加上京东商城左边的商品分类菜单了。

4.4 购物车

京东商城的商品购买有两个途径，一是在商品列表中单击具体的商品，进入商品详细信息页面，如果需要购买，就在这个详细信息页面单击购买按钮，该商品便加入到购物车中，页面跳转到购物车页面。另一种方式是直接在商品列表中单击购买按钮，跳过商品详细信息页面。考虑到商品详细信息页面实现起来相对简单，但是需要添加大量的商品信息，所以我这里只实现直接总商品列表中购买商品的功能。

大多数的购买行为不是只购买一件商品，所以我们需要允许用户继续购买，这样从程序员的角度来看就要解决如何暂时存放已经选购，但是还没有结账的商品。在这里有三个可能存放的方式，你先思考一下，有哪三个存放方式，它们的优缺点是什么？

第一种方式是将购物车信息存放在数据库专门的表中，好处是购物车内容将被持续存放，用户很有可能在选购的过程中，有意或是出现问题，浏览器突然关闭，这样的话再重新访问这个网站时，可以继续购物（要知道现在有人愿意尝试着购买），这对于购物网站来说就是巨大的财富。

这样我们需要定义一个购物车表，在其中存放用户 ID、购买商品的 ID，以及购买数量，至于其他的信息，比如商品名称、单价、描述等信息，完全可以再到商品表中查询。每次用户单击购买按钮后，我们向购物车表中写入这些信息，在购物车页面上，我们要根据用户 ID 来查询已购商品，并且显示出来。

存放到数据库中的缺点是，处理速度比较慢，数据库读写的速度一定慢于仅仅在内存中操作数据；很多用户再也不回来继续选购商品，会在数据库表中存放大量的垃圾数据；最关键的是这样做要求用户要先登录，因为购物车要绑定具体的用户才不会造成购物信息的混乱，但是我们知道大多数人在互联网上不喜欢登录这个环节，通常的思维习惯是先看看有没有自己想要的商品，如果有的话，会下意识的购买，只有商品已经吸引了用户，在结账的时候，该用户才不得不登录来购买，强制用户登录会挡住很多嫌麻烦的用户。

第二种方式，是将信息存放在 Session 中，我们知道 Session 是和浏览器关联的，也就是说 Session 能够识别正在访问的不同人，这样在购物过程中，就不必用户事先登录了。而且当用户在购物的过程中，如果不想买东西了，Session 会伴随着用户浏览器关闭而消失，我们不需要使用代码来维护这些垃圾信息，当然因为 Session 是内存中的集合对象，所以访问速度快

也是优点。

Session 只有两列，由于用户 ID 不需要存放，所以要存放的就是商品 ID 和购买数量，这样 Session 恰好可以满足我们的需求。问题是在很多时候，我们还会在 Session 中存放其他信息，不完全是购物车信息，我们不得不用代码来识别是商品信息，还是其他信息，由于编程的时候通常不知道其他信息是什么，所以筛选出来商品信息的程序不容易实现，因此通常我们会再定义一个专门存放购物车信息的集合，然后将购物车集合存放到 Session 中，仅仅利用 Session 和用户的关联性。

Session 的弱点是无法在用户关闭浏览器后，保存已经选购的商品，对于购物网站来说这将是一个很大的损失。

第三种方式是将信息存放到 Cookie 中，Cookie 也能够识别具体的某个人，比 Session 要好的是，Cookie 允许用户继续此前的购物过程。

使用 Cookie 存放购物车信息不容易，Cookie 也会涉及到商品信息和系统中的其他信息冲突的问题，Session 的办法在 Cookie 中不行，因为 Session 可以存放对象，这样我们就能够在 Session 中存放一个集合，而 Cookie 本质上看就是将字符串通过网络存放到客户端，没办法直接将集合存放到 Cookie 中，当然通过编码和解析我们也能在字符串和集合之间建立转换关系。Cookie 可能比数据库方式还要慢，因为通常网络速度要比硬盘的 IO 操作要慢。再有 Cookie 涉及隐私问题，一方面隐私有泄露的可能，另一方面用户可能禁用 Cookie，使得存放完全无法进行。

事实上现在的购物网站也并没有在使用哪个方案问题上达成一致，但是现在越来越有趋势结合使用多个存放方案来管理购物车，但通常会以 Session 为基础，根据用户登录或 Cookie 设置的情况来结合其他存放方案。我这里只选择使用 Session 来存放购物车。

我个人认为京东商城的购物流程并不是最佳方案，京东商城在单击购买按钮后，会跳转到一个专门的通知商品已成功加入购物车的页面，这时用户要选择是继续购物，还是去购物车结账。从商家的角度看，我们希望用户一次购买更多的商品，所以我认为单击购物按钮后，如果用户看到的还是商品展示列表，可能会促使用户产生更多的购买动作。当然一个好的购物网站并不全是技术问题，品牌、经营策略、商品丰富程度、价格、促销活动这些因素都比技术重要。

基于我的观念，我要实现的购物车不完全按照京东商城的网站来做，而是在用户单击购买按钮后，页面依然保持在商品展示上，页面的上面专门提供一个“去结账”的超链接，跳转到购物车。另外，我定义如果用户多次单击购买按钮，就意味着用户要购买多个数量的同一商品。

首先我们改造购买按钮，在单击这个按钮的时候，要页面跳转到自身，与此同时伴随着商品编号参数，具体实现可以在按钮基础上使用 onclick 事件，在事件中使用 JavaScript 代码跳转，也可以使用超链接，超链接使用一个按钮图片进行伪装。

```
<input type="button" value=" 购 买 " onclick="location.replace
('products.jsp?pid=<%= rs.getString(7)%>') ;">
```

要知道 pid 的值在提供的时候不要写成，"location.replace('products.jsp?pid="+ rs.getString(7)+

"%>') ;"，这是两个不同的语言。

在具体测试的时候，你会发现有问题，如果我们来到第 3 页，然后单击购买按钮，页面将自动跳转到第 1 页，原因很简单，在提供 pid 的时候，我们并没有提供 page，而 products.jsp 这个程序的逻辑是，如果没有找到 page，那么就默认提供第 1 页，但是我不想这样，我想停留在当前页面，看来最直接了当的办法是传递 pid 的同时，传递当前页码。

```
<input type="button" value=" 购 买 " onclick="location.replace('produc
ts.jsp?pid=<%= rs.getString(7)%>&page=<%=nowPage%>') ;">
```

如果你能够理解这行代码，我们继续，就在本页面，我们需要接收商品编号。

```
<%
String pid = request.getParameter("pid");
%>
```

可以打印出来看看有没有商品编号，下面我们要将这个 pid 放到购物车中，前面已经讨论过了，购物车是个集合，放在 Session 中的集合，对于一个用户，购物车只能创建一次，应该是在第一次单击购买按钮的时候创建的并且存入 Session，为了实现这样的效果，我们要判断一下，Session 中有没有购物车，如果没有就创建，如果有就从 Session 中取购物车集合的引用。

```
<!-- 购物车逻辑 -->
<%@ page import="java.util.*" %>
<%
String pid = request.getParameter("pid");
if(session.getAttribute("cart")==null){//还没有创建购物车
    HashMap cart = new HashMap();
    session.setAttribute("cart" , cart);
}

HashMap cart = (HashMap)session.getAttribute("cart");
```

准备好了购物车后，我们还不能将这个商品编号存入购物车，因为存在两种情况，第一次单击这个商品购买，和多次单击这个商品购买，我们先要判断一下，如果购物车中没有这个商品编号，那么就存入商品编号和数量 1，如果有这个商品编号，那么就要取出该商品的数量，然后加一再存入。

```
// 保存商品到购物车中
if(cart.get(pid)==null) {
    cart.put(pid , 1);
}else {
    int count = ((Integer)cart.get(pid)).intValue();
    cart.put(pid , count+1);
}
// 这是测试用语句，如果没有注释，在测试单击购买按钮的时候，会有信息打印到 Tomcat// 控
制台中
// System.out.println (cart);
```

我们在 menu.jsp 中加入超链接去结账

这样我们的任务转到 buy.jsp，还记得在 HTML 部分，我们实现过购物车的页面吗，有些部分代码是可以拿过来用的，只不过其中的商品部分要用程序生成。

下面是我准备的购物车基础页面，其中只有商品编号是用程序生成的，其余部分我都找了个东西占着表格的位置，等待着我一点点填充上去。

```
<%@ page contentType="text/html;charset=gb2312" %>
<%@ page import="java.util.*" %>
<html>
    <body>
        <table border="1">
            <tr bgColor="00ccff" align="center">
                <td width='7%' >商品编号</td>
                <td>商品名称</td>
                <td width='14%'>京东价</td>
                <td width='8%'>返现</td>
                <td width='8%'>赠送积分</td>
                <td width='9%'>商品数量</td>
                <td width='7%'>删除商品</td>
            </tr>
            <%
                HashMap cart = (HashMap)session.getAttribute(" cart") ;
                for(Object pid : cart.keySet()) {%>
                <tr align="center">
                    <td><%= pid %></td>
                    <td><a href=''><img src=''/>名称</td>
                    <td><font color="red">&yen;0.00</font></td>
                    <td>&yen;0.00</td>
                    <td>0</td>
                    <td><a href='' title='减一'>
<img src='img/bag_close.gif' border='none' /></a>
                     <input  type='text'  name='count'  maxleng  th='4'
size="1" value='1'/>
                     <a href='' title='加一'>
                        <img src='img/bag_open.gif'border='none' /></a>
                       </td>
                    <td><a href=''>删除</a></td>
                    </tr>
                <%} %>
        </table>
    </body>
</html>
```

看来有些信息还是要到数据库中查询才能知道，另外购买数量是从购物车中取得的。

```
<%@ page contentType="text/html;charset=gb2312" %>
<%@ page import="java.util.* , java.sql.*" %>
<% Class.forName("org.gjt.mm.mysql.Driver");
    Connection cn = DriverManager.getConnection
        ("jdbc:mysql://127.0.0.1:3306/360buy","root","123456");
```

```
 %>
<html>
   <body>
      <table border="1">
         <tr bgColor="00ccff" align="center">
            <td width='7%' >商品编号</td>
            <td>商品名称</td>
            <td width='14%'>京东价</td>
            <td width='8%'>返现</td>
            <td width='8%'>赠送积分</td>
            <td width='9%'>商品数量</td>
            <td width='7%'>删除商品</td>
         </tr>
         <%
            PreparedStatement ps = cn.prepareStatement
            ("select img ,name ,price from products where pid = ?");
            HashMap cart = (HashMap)session.getAttribute ("cart");

            float total = 0;// 汇总价格的变量
            for(Object pid : cart.keySet()) {
               ps.setString(1,(String)pid);
               ResultSet rs = ps.executeQuery();
               rs.next();
            total += rs.getFloat(3)*((Integer)cart.get(pid)).
intValue();
            %>
               <tr align="center">
                  <td><%= pid %></td>
                  <td><a href=''><img src='<%=rs.get String(1)%>'
                  width="30",height="30" border='none'align="top"/>
                  <%= changCode(rs.getString(2)) %> </td>
                  <td><font         color="red">&yen;<%=         rs.
getString(3) %>
</font></td>
                  <td>&yen;0.00</td>
                  <td>0</td>
                  <td><a href='' title='减一'>
<img src='img/bag_close.gif' border='none' /></a>
                  <input  type='text'  name='count'  ma  xlength='4'
size="1"
value='<%= cart.get(pid)%>'/>
                  <a href='' title='加一'>
<img src='img/bag_open.gif' border='none' /></a>
                  </td>
                  <td><a href=''>删除</a></td>
               </tr>
            <%}
          %>
```

```
                <tr>
                    <td colspan="7" align="right">
重量总计：1.25kg   原始金额：￥<%= total %>元 - 返现：￥0.00 元<br/>
商品总金额：￥<%= total %>元
                    </td>
                </tr>
    </table>
    </body>
    <%!
private String changCode(String s){
        String code = "";
        try {
            if(s!=null) {
                code = new String(s.getBytes("ISO-8859-1"), "GB2312");
            }
        }
    catch (Exception ex) {}
    return code;
}
 %>
</html>
```

汇总信息并没有做的和京东商城一样漂亮。还有一些信息我们设计的数据库表中没有，我就空缺下来，因为这些内容并不影响技术演练，如果我选择改表结构，并且添加足够的数据，可能会浪费太多篇幅，但是这并不是一个好的程序员应该有的态度，希望你能够尽可能地实现京东商城上面出现的所有功能。

4.4.1 实现加减按钮和删除商品的功能

现在我们要实现商品数量的增加、减少和删除功能。看起来加减就是两个小按钮，但是在我实现的过程中却发现非常困难，当用户单击按钮的时候，我们只能引发 JavaScript 事件，这样我们只能使用 JavaScript 代码，这时我遇到了第一个困难，找不到商品数量输入框中的值，我们知道如果找输入框的值，要有一个<form name="f1">，在此之前因为没有提交的必要，所以我没有准备<form>标记，但是添加了<form>标记发现使用 f1.count.value 还不行，我用了很长时间才发现，因为这个表格的内容是自动生成的，所以在 f1 的表单中，有很多 count，使用 alert 显示 f1.count 会发现结果不是 object，而是 object NodeList，结合起来看，NodeList 是合理的，只有这样 JavaScript 才能管理找到的多个 count。所以我用数组下标的形式来确定不同的输入框。

现在你通过我的分析尝试着自己实现，单击加减按钮，改变商品数量的值。

```
<script>
function cut(index,pid) {
    if(f1.count[index].value>1) {
        f1.count[index].value--;
    }
}
```

```
function add(index,pid) {
    f1.count[index].value++;
}
</script>
<!--在这里略去了一些 HTML 代码，主要是表头信息，以及数据库连接代码-->
<form name="f1">
<%
PreparedStatement ps = cn.prepareStatement
("select img ,name ,price from products where pid = ?");
HashMap cart = (HashMap)session.getAttribute("cart");
// 将做为参数传递到函数中，以便确定要修改那个 count 的值
int index = 0 ;
for(Object pid : cart.keySet()) {
    ps.setString(1,(String)pid);
    ResultSet rs = ps.executeQuery();
    rs.next() ;
%>
    <tr align="center">
    <td><%= pid %></td>
    <td><a href=''>
<img  src='<%=rs.getString(1)%>'  width="30",height="30"  border='none'
align="top"/>
    <%= changCode(rs.getString(2)) %></td>
    <td><font color="red">&yen;<%= rs.getString(3) %></font></td>
    <td>&yen;0.00</td>
    <td>0</td>
    <td><a href='javascript:cut(<%=index%>,<%=pid%>)' title='减一'>
<img src='img/bag_close.gif' border='none' /></a>
       <input type='text' name='count' maxlength='4' size="1"
value='<%= cart.get(pid)%>'/>
       <a href='javascript:add(<%=index%>,<%=pid%>)' title='加一'>
<img src='img/bag_open.gif' border='none' /></a>
    </td>
    <td><a href='buy.jsp?pid=<%=pid%>'>删除</a></td>
    </tr>
    <%index ++;
} %>
    <tr>
       <td colspan="7" align="right">
       重量总计：1.25kg   原始金额：￥
       元 - 返现：￥0.00 元<br/>
       商品总金额：￥元
    </td>
</tr>
</form>
```

虽然通过单击加减按钮能够改变商品数量的值，但是总金额并没有改变。另外购物车的商品数量并没有改变，如果不改变的话，将来结账的时候，就会出现数据不正确的情况，这个部分的难点

是，目前我们都是使用 JavaScript 来处理商品数量的值，而购物车必须使用 Java 代码来处理，没有直截了当的办法将 JavaScript 中的值发送到 Java 代码中，除非通过页面的参数传递，在此之前分页和购买商品的逻辑中，我们已经这样传递过参数了。

```
<script>
function cut(index,pid) {
   if(f1.count[index].value>1) {
      f1.count[index].value--;
location.replace("buy.jsp?pid="+pid+"&count="+f1.count[index].value);
   }
}
function add(index,pid) {
   f1.count[index].value++;
location.replace("buy.jsp?pid="+pid+"&count="+f1.count[index].value);
}
</script>
<!--在这里略去了一些 HTML 代码，主要是表头信息，以及数据库连接代码-->
<form name="f1">
<script>
   var total = 0;// 汇总货款
</script>
<%
PreparedStatement ps = cn.prepareStatement
("select img ,name ,price from products where pid = ?");
HashMap cart = (HashMap)session.getAttribute("cart");

// 加减按钮，改变购物车中的商品数量
String cc = request.getParameter("count");
if(cc!=null) {
   String addPid = request.getParameter("pid");
   cart.put(addPid, Integer.parseInt(cc));
}

int index = 0;
for(Object pid : cart.keySet()) {
   ps.setString(1,(String)pid);
   ResultSet rs = ps.executeQuery();
   rs.next();
%>
<script>
total += <%=rs.getFloat(3)*((Integer)cart.get(pid)).intValue()%>;
</script>
<tr align="center">
<td><%= pid %></td>
<td><a href=''>
<img  src='<%=rs.getString(1)%>'  width="30",height="30"  border='none'
align="top"/>
<%= changCode(rs.getString(2)) %></td>
```

```
<td><font color="red">&yen;<%= rs.getString(3) %></font></td>
<td>&yen;0.00</td>
<td>0</td>
<td><a href='javascript:cut(<%=index%>,<%=pid%>)' title='减一'>
<img src='img/bag_close.gif' border='none' /></a>
   <input type='text' name='count' maxlength='4' size="1" value='<%=
cart.get(pid)%>'/>
   <a href='javascript:add(<%=index%>,<%=pid%>)' title='加一'>
<img src='img/bag_open.gif' border='none' /></a>
</td>
<td><a href='buy.jsp?pid=<%=pid%>'>删除</a></td>
</tr>
<%index ++;
} %>
<tr>
   <td colspan="7" align="right">
重量总计：1.25kg   原始金额：￥
   <script>document.write(total)</script>
   元 - 返现：￥0.00 元<br/>
   商品总金额：￥<script>document.write(total)</script>元
   </td>
</tr>
</form>
```

当用户在商品数量的输入框中手工填写数量后，我们也要更改总金额，并且修改购物车。

下面是输入框事件定义的代码。

```
<input type='text' name='count' maxlength='4' size="1" value='<%=
cart.get(pid)%>' onblur="chang(<%=index%>,<%=pid%>)"/>
JavaScript 函数 chang()代码。
function chang(index, pid){
   location.replace("buy.jsp?pid="+pid+"&count="+f1.count[index].
value);
}
```

下面我们来实现删除的功能，这个相对简单，只要刷新页面的时候告诉下一个页面，从购物车中取出那个商品就行了。

删除链接的定义如下。

```
<td><a href='buy.jsp?pid=<%=pid%>&del=1'>删除</a></td>
```

删除逻辑代码如下。

```
// 删除逻辑
String del = request.getParameter("del");
if(del!=null) {
   String delPid = request.getParameter("pid");
   cart.remove(delPid);
}
```

这时你该会想到 AJAX，因为每单击一下加减按钮就要刷新一次页面看起来并不合理，我们完全可以用 AJAX 将单击的操作发送到一个专门的程序中，这个程序会修改购物车，页面上的变化可以使用 JavaScript 来维护，你可以试着自己用 AJAX 来重新实现一遍。

第 5 章

使用 JavaBean

相信你也已经能够感受到，我们仅仅是添加了看起来并不起眼的功能，但程序逻辑的难度好像是增加了一倍还不止，JSP 的方案在设计之初看起来很好，将程序员从严格的 Java 语法中摆脱了出来，但是当程序复杂到了一定程度后，我们发现搅在一起的 HTML、Java、JavaScript，还有 SQL 代码，让这个页面成了一团乱麻，最后会因为牵一发而动全身，使得程序几乎无法维护。

每当到这个时候，就会出现技术将程序的不同部分分离到不同的文件中，这样就可以让不同专长的人合作来实现一个程序，此前这些搅在一起的程序要想多人合作太难了。

事实上我们已经经历过一些分离了，至少 HTML、JavaScript 和 CSS 完全可以分离到不同的文件中，我们其实正在享受这三类技术分离的成果，但是现在问题的焦点在 HTML 和 Java 需要分离，要知道专业做 HTML 的人和专业做 Java 的人从本质上完全不同，要求程序员同时做到这两个技术都很专业，是强人所难，一个偏重于美工，一个偏重于逻辑，所以现在迫切的需要将 HTML 和 Java 分离开。这可真有意思，JSP 不就是 HTML+Java 产生的吗？

5.1 使用 JavaBean 实现用户验证

5.1.1 定义 JavaBean

分离使用的技术是 JavaBean，加入了 JavaBean 的开发模式也叫做 JSP+JavaBean。什么是 JavaBean，事实上 JavaBean 就是一个普普通通的类，JavaBean 只是代表着一个思想，一个分离的思想，有些对 JavaBean 特别的定义，但是并不十分严格，更新的语言在类似技术定义上已经比 Java 严格了。

现在我们看代码，看 JSP 和 JavaBean 是如何结合的。在 JavaBean 中有一个概念叫做属性，理论上讲属性不是成员变量，也不是成员方法，从类的角度看属性是成员方法，但是主要行使的是成员变量的功能。下面是一个典型 JavaBean 的属性定义。

```
package com.wy.bean;

public class UserBean {
    private String user;
    private String pass;
```

```
    public void setUsername(String value) {
        user = value;
    }
    public String getUsername() {
        return user;
    }
    public void setPass(String value) {
        pass = value;
    }
}
```

上面的 JavaBean 代码不是常用的代码，用这个代码我们来认清一些概念，我们看到了三个方法 setUsername，getUsername，setPass，那么我们说这个 JavaBean 有两个属性，一个是 username，一个是 pass，注意方法中 Username 的 U 是大写的，这是作为属性必须的，当我们称其为属性时，u 是小写的。存放值的变量声明成 user，你要清楚，属性和成员变量是什么无关，它们只和方法名字有关，当然在现实的代码中，我们通常声明的成员变量和属性相同，没有必要起很多名字而已。在这个 JavaBean 的属性中，username 是可读可写的属性，而 pass 是可写不可读的属性，因为它没有 GET 方法。

现在你在 UserBean 中加入属性 pass 的可读方法，然后我们用这个 JavaBean 来改造 loginc.jsp。

```
<%@page contentType="text/html;charset=GB2312"%>
<%@page import="java.sql.*%>
<jsp:useBean id="ub" class="com.wy.bean.UserBean"/>
<jsp:setProperty name="ub" property="username" param="username"/>
<jsp:setProperty name="ub" property="pass" param="password"/>
<%
    Class.forName("org.gjt.mm.mysql.Driver");
    Connection cn = DriverManager.getConnection
("jdbc:mysql://127.0.0.1:3306/360buy","root","123456");
    PreparedStatement ps = cn.prepareStatement
("select * from user where username=? and password=?");
    ps.setString(1, ub.getUsername());
    ps.setString(2, ub.getPass());
    ResultSet rs = ps.executeQuery();
    if(rs.next()) {
        session.setAttribute("user", ub.getUsername());

        Cookie username = new Cookie("user", ub.getUsername());
        username.setMaxAge(60*60*24*365);
        response.addCookie(username);
        %>
        <jsp:forward page="products.jsp"/>
    <%}else {%>
        <jsp:forward page="reg.jsp"/>
    <%}
 %>
```

在测试时，别忘了要将 UserBean 编译，并部署到 classes 目录中，我想你应该能够得出一个结论，类都应该放到 classes 目录中，然后要重启 Tomcat。改造 loginc.jsp 后，你应该能够看到新的 loginc.jsp 可以正常工作了。

现在我们再来看一下我们改造了什么？<jsp:useBean id = "ub" class = "com.wy.bean.User Bean"/>，这句话的作用就是创建 UserBean 的对象，作用类似于 com.wy.bean.UserBean ub = new com.wy.bean.UserBean() ;，但是你不能用后面的这个 Java 代码来替代。

后面的两句话将 login.jsp 中表单的值传入 UserBean 的属性中，新的 JSP 标记替代了 request.getParameter 方法。还记得我们的目标是什么？消除 Java 代码，这三句话用的可不是 Java 语法。

<jsp:setProperty name="ub" property="pass" param="password"/>，这句话说明了如果表单中的名称和 JavaBean 的属性名字不同该怎么办，通过这个例子我们很清楚 property 的值是 JavaBean 中的属性名字，而 param 的值指的是表单的名字。

在后面 Java 代码中，可以直接使用 JavaBean 类中成员方法的写法。事实上还有一种办法读取 JavaBean 中的属性值。<jsp:getProperty name="ub" property="username"/>，只是这样写不容易和 Java 代码融合，所以相比而言用的并不是太多。

到目前为止，根本看不出使用 JavaBean 的好处，我们要剔除掉 Java 代码的目标没有实现，甚至连节省代码量都没看出来。事实上还有一种简写的形式，我们要改造一下 JavaBean。

```
package com.wy.bean;

public class UserBean {
    private String user;
    private String pass;
    public void setUsername(String value) {
        user = value;
    }
    public String getUsername() {
        return user;
    }
    public void setPassword(String value) {
        pass = value;
    }
    public String getPassword() {
        return pass;
    }
}
```

我将属性的名字改成和 login.jsp 的表单项目相同的名字，那么使用 JavaBean 的前三句话就变成下面的形式了。

```
<jsp:useBean id="ub" class="com.wy.bean.UserBean"/>
<jsp:setProperty name="ub" property="*"/>
```

当 setProperty 的 property 属性设置为*时，Tomcat 会进行自动的匹配，将 JavaBean 属性名字和表单名字吻合的项目匹配到一起去，值也将被适当的传递。

趁着这个机会我们来思考一个问题，我们知道同一个任务 Servlet 和 JSP 都可以实现，当然设计之初 JSP 是为了替代 Servlet 的，但是体验到现在，你觉得像 loginc 这样的程序是用 Servlet 合适，还是用 JSP 合适呢？我们很容易就得出结论，如果 Java 代码多，那么 Servlet 比较合适，如果 HTML 代码多，那么 JSP 合适，所以在开发的时候，就有了 Servlet 模式、JSP 模式和 JSP+Servlet 模式，而现在我们来探索的是 JSP+JavaBean 的模式，这么说来 JavaBean 在某种程度上，有替代 Servlet 的意思。

5.1.2 运用 JavaBean

我们继续来看代码，我们知道一旦在 JSP 页面上对 JavaBean 进行了传值，那么事实上用户名和密码的值就会传递到 UserBean 的成员变量 user 和 pass 中，既然这样我们何不直接在 UserBean 中写代码进行用户身份验证呢。具体代码如下。

```
package com.wy.bean;

import java.sql.*;

public class UserBean {
    private String user;
    private String pass;
    public void setUsername(String value) {
        user = value;
    }
    public String getUsername() {
        return user;
    }
    public void setPassword(String value) {
        pass = value;
    }

    //用户身份验证方法
    public boolean vali() {
        boolean b = false;
        try {
            Class.forName("org.gjt.mm.mysql.Driver");
            Connection cn = DriverManager.getConnection
("jdbc:mysql://127.0.0.1:3306/360buy","root","123456");
            PreparedStatement ps = cn.prepareStatement
("select * from user where username=? and password=?");
            ps.setString(1,user);
            ps.setString(2,pass);
            ResultSet rs = ps.executeQuery();
            b = rs.next();
```

```
        }
        catch (Exception ex) {}
        return b;
    }
}
```

代码非常容易理解，要提请注意的是，getPassword 方法被我取消了，因为确实没用，而 getUsername 保留，将用户名保存到 Session 中，以及保存到 Cookie 的逻辑中，需要取得用户名。这样 loginc.jsp 就可以改造成如下代码。

```
<%@page contentType="text/html;charset=GB2312"%>
<jsp:useBean id="ub" class="com.wy.bean.UserBean"/>
<jsp:setProperty name="ub" property="*"/>
<%
    if(ub.vali()) {
        session.setAttribute("user" , ub.getUsername());

        Cookie username = new Cookie("user" , ub.getUsername());
        username.setMaxAge(60*60*24*365);
        response.addCookie(username);
        %>
        <jsp:forward page="products.jsp"/>
    <%}else {%>
        <jsp:forward page="reg.jsp"/>
    <%}
 %>
```

需要提示的是，在测试时，可能需要刷新一下页面，才能在商品展示页中显示正确的用户名信息。

我们的代码越来越体现出 JavaBean 的意义来了，数据库访问的逻辑都被包装到了 JavaBean 中，如果将获取 Connection 在提炼到数据库连接的工厂类中就好了，如果剔除门户之别，我们完全可以直接使用之前我们做的那个 DataBase 的 Servlet，这样就成了 JSP+JavaBean+Servlet 结构了。

到目前我们发现所有和 Web 程序相关的 Java 代码没办法放到 JavaBean 中，JavaBean 帮我们分离的通常是传统的 Java 程序的逻辑。

5.1.3　JavaBean 的作用域

事实上我们可以将 session.setAttribute("user" , ub.getUsername());分离出去，改造代码如下。

```
<%@page contentType="text/html;charset=GB2312"%>
<jsp:useBean id="ub" class="com.wy.bean.UserBean"  scope="session"/>
<jsp:setProperty name="ub" property="*"/>
<%
    if(ub.vali()) {
        Cookie username = new Cookie("user", ub.getUsername());
        username.setMaxAge(60*60*24*365);
```

```
            response.addCookie(username);
            %>
            <jsp:forward page="products.jsp"/>
        <%}else {%>
            <jsp:forward page="reg.jsp"/>
        <%}
    %>
```

将<jsp:useBean>标记的 scope 属性设置成 session，JavaBean 默认的生命周期是 page，也就是在当前页面有效，这样设置了，这个 JavaBean 将在整个 session 有效的情况下，保存在 session 中，看起来读取用户名的代码也要有所修改，现在读取用户的代码在 menu.jsp 中。改造后的代码如下。

```
<jsp:useBean id="ub" class="com.wy.bean.UserBean" scope="session"/>
<%if(ub.getUsername()==null){ %>
    <li>您好！欢迎来到京东商城！</li>
<%        }else{ %>
    <li>你好，<jsp:getProperty name="ub" property="username"/>!</li>
<% }%>
```

在看上面代码时，要知道工作思路是，从 session 中取出 UserBean 的对象 ub，然后是使用 ub 的方法获取用户名。

我很好奇直接用 Java 代码来取 session 中的对象行不行。

```
<%com.wy.bean.UserBean ub=(com.wy.bean.UserBean)session.getAttribute("ub") ;
if(ub.getUsername()==null){ %>
    <li>您好！欢迎来到京东商城！</li>
<%        }else{ %>
    <li>你好，<%= ub.getUsername() %>!</li>
<% }%>
```

结果这个代码也出来正确的效果，这充分证明了 UserBean 就是以 ub 的名字存放在 Session 中。

5.1.4 在 JavaBean 中使用内置对象

下面我们将添加 Cookie 的代码也封装到 JavaBean 中，这时遇到的问题是，创建 Cookie 对象不难，但是最后将 Cookie 送到客户端，需要使用 response 对象，而我们的 JavaBean 没有这些内置对象，只有 JSP 中有 response 对象，看来只能设置一个接受参数，将 JSP 中的 response 传递到 JavaBean 中。

```
package com.wy.bean;

import java.sql.*;
import javax.servlet.http.*;
public class UserBean {
    private String user;
    private String pass;
    public void setUsername(String value) {
```

```
        user = value ;
    }
    public String getUsername() {
        return user;
    }
    public void setPassword(String value) {
        pass = value;
    }

    //用户身份验证方法
    public boolean vali(HttpServletResponse response) {
        boolean b = false;
        try {
            Class.forName("org.gjt.mm.mysql.Driver");
            Connection cn = DriverManager.getConnection
("jdbc:mysql://127.0.0.1:3306/360buy","root","123456");
            PreparedStatement ps = cn.prepareStatement
("select * from user where username=? and password=?");
            ps.setString(1,user);
            ps.setString(2,pass);
            ResultSet rs = ps.executeQuery();
            b = rs.next();
            if(b){
                addCookie(response);
            }
        }
        catch (Exception ex) {}
        return b;
    }
    private void addCookie(HttpServletResponse response) {
        Cookie username = new Cookie("user" , user);
        username.setMaxAge(60*60*24*365);
        response.addCookie(username);
        System.out.println (username);
    }
}
```

你看现在的 JavaBean 是越来越像 Servlet 了，现在的 loginc.jsp 变成下面的样子。

```
<%@page contentType="text/html;charset=GB2312"%>
<jsp:useBean id="ub" class="com.wy.bean.UserBean" scope="session"/>
<jsp:setProperty name="ub" property="*"/>
<%if(ub.vali(response)) {%>
    <jsp:forward page="products.jsp"/>
<%}else {%>
    <jsp:forward page="reg.jsp"/>
<%} %>
```

想一下，将 Java 代码从 JSP 中分离出去意味着什么？意味着 JSP 变得简洁多了，这样就可以专注地处理显示的界面，也意味着可以专注地处理 Java 逻辑。

5.2 使用 JavaBean 来实现商品展示

5.2.1 规划和设计 JavaBean

我们已经实现多遍商品展示的页面了，其中的流程应该相当的清楚，这样我们就能够事先规划和设计 JavaBean。JavaBean 将包含两个方面的功能，一是需要什么属性，二是需要提供什么给页面。

再来回顾一下商品展示页面的设计思路，其中依照上下页传递的参数来设置 JavaBean 的属性，明确的有用于分页的 nowPage 和用户购物的 pid，pid 属性是不可读的，你思考一下，什么时候会传 pid 的参数？是单击购物按钮的时候，所以写入的 pid 属性事实上是购物车逻辑。但是在购物车的界面上，我们要显示已购商品的信息，所以需要返回购物车内容，我使用了方法 getCart。

在页面显示内容中，我们需要总页码信息 pageCount，这个属性为只读属性。为了让 JavaBean 更强大，以便适应页面功能的进一步提升，我们还将提供 pageSize 属性，由于本案例的每页显示商品数量取决于行数和每行显示的商品数量，所以不但提供了 pageSize 属性，还提供了 cells 和 rows 属性。

而页面需要的信息主要是显示的商品信息。

从这个任务开始，我们将数据库连接的 Connection 对象获取放到 DataBase 中去，具体的代码参照 Servlet 部分的内容。

JavaBean 代码如下。

```
package com.wy.bean;

import java.sql.*;
import com.wy.DataBase;
import java.util.*;

public class Products {
    private int nowPage = 1;
    private HashMap<String, Integer> cart = new HashMap<String, Integer>();
    private int cells = 4;
    private int rows = 2;

    public void setNowPage(int value){
        nowPage = value;
    }
    public int getNowPage() {
        return nowPage;
    }

    public void setCells(int value){
        cells = value;
```

```
    }
    public int getCells(){
        return cells;
    }

    public void setRows(int value){
        rows = value;
    }
    public int getRows(){
        return rows;
    }

    public int getPageSize() {
        return cells*rows;
    }

    // 购物车逻辑
    public void setPid(String value){
        if(cart.get(value)==null){
            cart.put(value , 1);
        }else {
            cart.put(value , cart.get(value)+1);
        }
}

    // 获取购物车
    public HashMap getCart() {
        return cart;
    }

    // 获取总页码
    public int getPageCount() {
        int count = 0;
        int rowCount = this.rowCount();
        if(rowCount%this.getPageSize()==0){
            count = rowCount/this.getPageSize();
        }else {
            count = rowCount/this.getPageSize() + 1;
        }
        return count;
    }
    // 获取商品总数
    private int rowCount() {
        int count = 0;
        try {
            Connection cn = DataBase.getConnection();
            Statement st = cn.createStatement();
            ResultSet   rs   =   st.executeQuery("select   count(*)   from
```

```
products");
            rs.next();
            count = rs.getInt(1);
        }
        catch (Exception ex) {}
        return count;
    }

    // 获取商品信息
    public ResultSet getProducts() {
        ResultSet rs = null;
        try {
            Connection cn = DataBase.getConnection();
            Statement st = cn.createStatement();
            rs = st.executeQuery
("select img, name, price, gift, down, info, pid from products");

            //跳过
            for (int i=0; i<(nowPage-1)*this.getPageSize(); i++){
                rs.next();
            }
        }
        catch (Exception ex) {}
        return rs;
    }
}
```

上面的代码对于你来说应该是相当的好理解，将这个 JavaBean 部署到 classes 中以后，我们要写 JSP 代码。

5.2.2 改造 JSP

虽然题目是改造 JSP，我建议你还是重新写一遍 JSP 吧，这样能够体会使用 JavaBean 的喜悦。

```
<html>
<body>
<jsp:include page="menu.jsp"/>
<!-- 准备 -->
<%@ page contentType="text/html;charset=gb2312" %>
<%@ page import="java.sql.*" %>

<jsp:useBean id="pro" class="com.wy.bean.Products"/>
<jsp:setProperty name="pro" property="*"/>
<!-- 显示当前记录 -->
<table width="100%">
<%
    Connection cn = com.wy.DataBase.getConnection();
```

```
   ResultSet rs = pro.getProducts();
   PreparedStatement ps = cn.prepareStatement
("select count(pid) from comment where pid=?");
   for(int i = 0; i < pro.getRows(); i ++) {%>
      <tr align="center">
      <% for(int j = 0; j < pro.getCells(); j ++) {
         if(rs.next()){
         %>
         <td>
         <!-- 图片 -->
         <img src="<%=rs.getString(1)%>"/>
         <!-- 商品名称 -->
         <div id="name">
            <a href="" title="<%=changCode(rs.getString(6))%>">
<%= changCode(rs.getString(2)) %></a>
         </div>
         <!-- 显示单价 -->
         <div class='price'>&yen;<%= rs.getString(3) %></div>

         <!-- 显示评论 -->
         <%
         ps.setInt(1, rs.getInt(7));
         ResultSet crs = ps.executeQuery();
         crs.next();
          %>
         已有<%= crs.getString(1) %>人评论

         <!-- 显示赠品图标 -->
         <%    if(rs.getInt(4) !=0){%>
            <a class="p1" title='购买本商品送赠品'><img src= 'img/gift.
png'></a>
         <%} %>

         <!-- 显示直降图标 -->
         <%    if(rs.getString(5).equals("y")){%>
               <a class="p2" title='本商品正在降价销售中'>
<img src='img/down.png'></a>
         <%}%>

         <!-- 显示按钮 -->
         <div>
         <input type="button" value="购买"
onclick="location.replace('products.jsp?pid=<%=
rs.getString(7)%>');">
         <input type="button" value="关注">
         <input type="button" value="对比">
         </div>
         </td>
```

```
        <%}
        }%>
        </tr>
    <%}
 %>
</table>
<!-- 声明成员方法 -->
<%!
private String changCode(String s){
    String code = "";
    try {
        if(s!=null) {
            code = new String(s.getBytes("ISO-8859-1"), "GB2312");
        }
    }
    catch (Exception ex) {}
    return code;
}
 %>

<!-- 显示页码 -->
<% for(int i = 1; i <= pro.getPageCount(); i ++) {%>
    <a href="products.jsp?nowPage=<%= i%>"><%= i %></a>
<%} %>
</body>
</html>
```

已经能够看出 JSP 的代码大幅度地减少了，过去的代码 145 行，现在的只有 82 行。

但是还有很多 Java 代码在 JSP 文件中，我们发现这些代码大多和显示相关，JavaBean 虽然也能使用传递来的 response 输出显示结果，但是毕竟 JavaBean 对于显示并不擅长。还有一些不得不写的数据库访问和逻辑代码，这是我们要解决掉的问题。

5.2.3 将数据库和页面彻底分离开

将数据库和页面分离开的理由有几个，我们看到的商品信息是通过 JavaBean 的方法提供的，提供的形式是 ResultSet，这就带来了一个问题，我们什么时候将数据库的连接关闭，在 JSP 中是不可能关闭的，在 JSP 文件中，压根就找不到 Connection 对象的引用，而在 JavaBean 的方法中更不能关闭数据库连接了，因为 ResultSet 到达页面还要用，如果在 JavaBean 中关闭了 Connection，那么 ResultSet 中也找不到值了。

从软件设计的高度看，我们通常能够将软件分成界面、程序逻辑和数据库访问，当软件项目很大的时候，清晰地将各个部分分离开有利于未来的维护和扩展，这也支持我们将页面中访问数据库的代码分离开，最后形成页面只和 JavaBean 打交道，JavaBean 和数据库打交道。

理论归理论，具体到我们这个任务中，该如何实现分离呢？如果不将 ResultSet 传递到页面，

那么又有什么能够装载那么多数据进行传递呢？现在看只有集合能完成这样的使命，但是集合只有一列，最多两列，而我们所传递的数据绝不只两列，就是一个二维表，过去我们建立二维表集合的做法是，在集合列中放入集合，但是这样做的问题是没办法识别存入的信息是什么。

现在我们学到了 JavaBean，完全可以将 JavaBean 和集合结合起来，实现更加实用的二维表，先准备 ProBean，用来存放一个商品。

```
package com.wy.bean;

public class ProBean {
    private String pid;
    private String name;
    private String img;
    private float price;
    private int gift;
    private String down;
    private String info;
    private int comment; public String getPid() {
        return pid;
    }
    public void setPid(String pid) {
        this.pid = pid;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getImg() {
        return img;
    }
    public void setImg(String img) {
        this.img = img;
    }
    public float getPrice() {
        return price;
    }
    public void setPrice(float price) {
        this.price = price;
    }
    public int getGift() {
        return gift;
    }
    public void setGift(int gift) {
        this.gift = gift;
    }
    public String getDown() {
```

```
        return down;
    }
    public void setDown(String down) {
        this.down = down;
    }
    public String getInfo() {
        return info;
    }
    public void setInfo(String info) {
        this.info = info;
    }
    public int getComment() {
        return comment;
    }
    public void setComment(int comment) {
        this.comment = comment;
    }
}
```

上面的这个 JavaBean 代码毫无技术含量，就是将页面需要的商品信息变成对应的属性，你发现这些属性是和需求对应的，不是和数据库对应的，其中的 comment 属性就不是商品数据库中的列，我们知道这个属性存放的是评论数量。在 Eclipse 中有工具，在你提供了变量后，便会生成给你 get 和 set 代码。

下面我来提供修改后的 getProducts 方法。

```
// 获取商品信息
public HashSet getProducts() {
    HashSet pro = new HashSet();
    try {
        Connection cn = DataBase.getConnection();
        Statement st = cn.createStatement();
        ResultSet rs = st.executeQuery
("select img, name, price, gift, down, info, pid from products");

        //跳过
        for (int i = 0; i<(nowPage-1)*this.getPageSize(); i++){
            rs.next();
        }

        // 获取评论数量
        PreparedStatement ps = cn.prepareStatement
("select count(pid) from comment where pid=?");
        //将信息保存到集合中
        for (int i = 0; i<this.getPageSize(); i++){
            if(rs.next()){
                ProBean pb = new ProBean();
                pb.setImg(rs.getString(1));
                pb.setName(changCode(rs.getString(2)));
```

```
                pb.setPrice(rs.getFloat(3));
                pb.setGift(rs.getInt(4));
                pb.setDown(rs.getString(5));
                pb.setInfo(changCode(rs.getString(6)));
                pb.setPid(rs.getString(7));

                ps.setString(1,rs.getString(7));
                ResultSet rs1 = ps.executeQuery();
                rs1.next();
                pb.setComment(rs1.getInt(1));

                pro.add(pb);
            }
        }

        rs.close();
        st.close();
        cn.close();
    }catch (Exception ex) {}
    return pro;
}
// 转换从数据库中取值时遇到的中文编码
private String changCode(String s){
String code = "";
try {
        if(s!=null) {
            code = new String(s.getBytes("ISO-8859-1") , "GB2312");
    }
}catch (Exception ex) {}
return code;
}
```

可以看出来，为了在提供数据的时候避免提供 ResultSet，我们将每一行数据包装到了一 个 ProBean 对象中，然后将对象存入 HashSet 集合中，这样操作完毕，就可以将数据库资源都关闭掉了，而将集合提供给页面。

这里为了清晰地说明问题，我将关闭代码放到 try 语句中，事实上应该放到 finally 语句中。

这样我们的 JSP 文件就要做出相应的修改。

```
<html>
<body>
<jsp:include page="menu.jsp"/>
<!-- 准备 -->
<%@ page contentType="text/html;charset=gb2312" %>
<%@ page import="java.util.*,com.wy.bean.ProBean" %>

<jsp:useBean id="pro" class="com.wy.bean.Products"/>
<jsp:setProperty name="pro" property="*"/>
<!-- 显示当前记录 -->
```

```
<table width="100%">

<%
    //获取数据
    HashSet<ProBean> pbs = pro.getProducts();
    int cell = 0 ;//用于管理输出<tr></tr>

    for(ProBean pb : pbs) {
        if(cell%pro.getCells()==0) {%>
            <tr align="center">
        <% }
        cell ++;%>
        <td>
        <!-- 图片 -->
        <img src="<%=pb.getImg()%>"/>
        <!-- 商品名称 -->
        <div id="name">
            <a href="" title="<%=pb.getInfo()%>"><%= pb.getName() %></a>
        </div>
        <!-- 显示单价 -->
        <div class='price'>&yen;<%= pb.getPrice() %></div>

        <!-- 显示评论 -->
        已有<%= pb.getComment() %>人评论

        <!-- 显示赠品图标 -->
        <%       if(pb.getGift() !=0){%>
            <a  class="p1"  title=' 购 买 本 商 品 送 赠 品 '><img  src='img/
gift.png'></a>
        <%} %>

        <!-- 显示直降图标 -->
        <%       if(pb.getDown().equals("y")){%>
               <a class="p2" title='本商品正在降价销售中'>
               <img src='img/down.png'></a>
        <%}%>

        <!-- 显示按钮 -->
        <div>
        <input type="button" value="购买"
onclick="location.replace('products.jsp?pid=<%= pb.getPid()%>') ;">
        <input type="button" value="关注">
        <input type="button" value="对比">
        </div>
        </td>
        <%if(cell%pro.getCells()==0) {%>
            </tr>
        <%}
```

```
    }
        %>
</table>

<!-- 显示页码 -->
<% for(int i = 1; i <= pro.getPageCount(); i ++) {%>
    <a href="products.jsp?nowPage=<%= i%>"><%= i %></a>
<%} %>
</body>
</html>
```

这些代码不会十分顺利的实现，或许你大量的精力都要用于消除页面上出现的错误信息，如果错误没有了，但是页面显示的和想象不同，还要通过 System.out.println();将自己关心的信息显示到 Tomcat 控制台上。

这些代码其实有个问题，你刷新一下页面看看，商品的排列顺序不稳定，过去直接从数据库中取数据受到数据库表的排列循序约束，现在转换到了集合中，这反倒成了一个问题，或许我们应该使用 ArrayList，你试着自己修改一下代码，看看问题是否能够解决。

虽然我之前提到过技术一步步发展到现在的原因是什么，但是想必你还是越来越疑惑，现在完成相同的任务代码越来越多，过去一个文件就解决的问题，现在却需要好几个文件，尤其是数据库访问和页面分离，几乎就是费了两遍事。

不知道你是否能够体会到，这些变化的背后包含了一个主线，就是代码不断的分离，分离的好处是弱耦合，什么是弱耦合？城市里两节连在一起的公交车和火车相比，公交车是强耦合，你不能简单地将公交车的两节分开或是合并，而火车做这些事情就便利多了。

我们的程序具有同样道理，最早的将所有代码都写到一起的 JSP 文件，代码的每个部分之间就是强耦合关系，你不能随便的改动，因为会影响到其他部分，更不要说代码重用，单元测试，团队协作了。

现在我们看起来代码复杂多了，但是代码分离了，弱耦合了，每部分代码都可以单独测试确保没有问题，对一个部分代码的修改不用担心会影响其他部分，积累时间久了，一些程序可以稍加修改，甚至稍加设置便用到其他项目去了，其实在同一个项目中也能找到很多这样的重用，因为弱耦合所以一个任务可以分给不同的人来完成，而且每个人工作的划分和技术相关，这样使用不同技术的人就会越来越专业，也有利于代码质量的提高，如果人员发生了变动，对团队而言，伤害也会小得多。

5.3　实现购物车逻辑

在上面的代码中，我们使用了两个 JavaBean，一个是非常简单的，纯粹的用于存放数据　的 JavaBean，而另一个中包含了很多逻辑代码，以后你会遇到数据 Bean 和业务 Bean 这类的称呼，虽然我们这里不能这样称呼，但是我们开始渐渐的体验它们真正的含义。

在 Products.java 中，我们有两个方法，一个是 setPid，和其他的属性不同，这个方法将得到的 pid 放到了集合中，这个集合就是我们之前使用的购物车，没有对应的 getPid，因为我们要得到的是存放了很多商品信息的集合，所以如果一定要说是哪个方法和 setPid 对应，就是 getCart 了。

在此前的代码中，getCart 只是简单地将集合 cart 返回，我们知道 cart 中存放的是商品 pid 和对应的购买数量，可是购物车信息显示的时候需要更多的信息，过去的做法是利用集合中的 pid 再次到数据库中查询其他信息，这就意味着在购物车的页面中，我们需要访问数据库，我们已经开始将数据库的访问和页面分离开了，所以这里也需要这样做，我的思路是改造 getCart 方法，在这个方法中就访问数据库，得到所有需要的信息后，通过集合将商品信息提供到页面上，问题是这个集合中存放的已购商品的信息和商品展示时的信息有所不同，所以我们要准备一个已购商品的 JavaBean 来存放不同的信息。

```
package com.wy.bean;

public class BuyBean {
    private String pid;
    private String name;
    private float price;
    private float reMoney;
    private int accum;
    private int count;

    public String getPid() {
        return pid;
    }
    public void setPid(String pid) {
        this.pid = pid;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public float getPrice() {
        return price;
    }
    public void setPrice(float price) {
        this.price = price;
    }
    public float getReMoney() {
        return reMoney;
    }
    public void setReMoney(float reMoney) {
        this.reMoney = reMoney;
    }
```

```
    public int getAccum() {
        return accum;
    }
    public void setAccum(int accum) {
        this.accum = accum;
    }
    public int getCount() {
        return count;
    }
    public void setCount(int count) {
        this.count = count;
    }
}
```

下面我们要改造 getCart 方法了，另外为了得到总的货款信息，以及支持购买数量改变和删除商品，我们还要提供一些属性。

```
private float total;
private String id;
// 获取购物车信息
public ArrayList<BuyBean> getCart() {
    ArrayList<BuyBean> buy = new ArrayList<BuyBean>();
    try {
        total = 0;
        Connection cn = DataBase.getConnection();
        PreparedStatement ps = cn.prepareStatement(
"select img ,name ,price from products where pid = ?");
        for(String pid : cart.keySet()){
            ps.setString(1 , pid);
            ResultSet rs = ps.executeQuery();
            rs.next();

            BuyBean bb = new BuyBean();
            bb.setPid(pid);
            bb.setImg(rs.getString(1));
            bb.setName(changCode(rs.getString(2)));
            bb.setPrice(rs.getFloat(3));
            bb.setReMoney(0);
            bb.setAccum(0);
            bb.setCount(cart.get(pid));

            total += bb.getPrice() * bb.getCount();

            buy.add(bb);
        }
    }
    catch (Exception ex) {}
    return buy;
}

public float getTotal() {
```

```
    return total;
}

// 改变购买数量
public void setId(String id){
    this.id = id;
}
public void setCount(int count){
    cart.put(id , count);
}

// 删除逻辑
public void setDel(String id){
    cart.remove(id);
}
```

重新实现 buy.jsp 代码。别忘了在这之前，将 products.jsp 中<jsp:useBean>的作用域声明成 session。

```
<%@ page contentType="text/html;charset=gb2312" %>
<%@ page import="java.util.* , java.sql.* , com.wy.bean.BuyBean" %>

<jsp:useBean id="pro" class="com.wy.bean.Products" scope="session"/>
<jsp:setProperty name="pro" property="*"/>

<script>
function cut(index,pid) {
    if(f1.count[index].value>1) {
        f1.count[index].value--;
        location.replace("buy.jsp?id="+pid+"&count="+f1.count[index].
value);
    }
}
function add(index,pid) {
    f1.count[index].value++;
    location.replace("buy.jsp?id="+pid+"&count="+f1.count[index].value);
}
function chang(index, pid){
    location.replace("buy.jsp?id="+pid+"&count="+f1.count[index].value);
}
</script>

<html>
    <body>
        <table border="1">
        <form name="f1">
            <tr bgColor="00ccff" align="center">
                <td width='7%' >商品编号</td>
                <td>商品名称</td>
                <td width='14%'>京东价</td>
                <td width='8%'>返现</td>
                <td width='8%'>赠送积分</td>
```

```
                <td width='9%'>商品数量</td>
                <td width='7%'>删除商品</td>
            </tr>
            <%
                ArrayList<BuyBean> buy = pro.getCart();

                int index = 0;
                for(BuyBean bb : buy) {%>

                <tr align="center">
                    <td><%=bb.getPid() %></td>
                    <td><a href=''><img src='<%=bb.getImg()%>' width="30",
height="30" border='none' align="top"/>
                    <%= bb.getName() %></td>
                    <td><font color="red">&yen;<%=bb.getPrice() %>
</font></td>
                    <td>&yen;<%= bb.getReMoney() %></td>
                    <td><%= bb.getAccum() %></td>
                    <td>
<a href='javascript:cut(<%=index%>,<%=bb.getPid()%>)' title='减一'>
<img src='img/bag_close.gif' border='none' /></a>
<input type='text' name='count' maxlength='4' size="1" value='<%= bb.
getCount()%>'
onblur="chang(<%=index%>,<%=bb.getPid()%>)"/>
 <a href='javascript:add(<%=index%>,<%=bb.getPid()%>)' title='加一'>
<img src='img/bag_open.gif' border='none' /></a>
                </td>
                <td><a href='buy.jsp?del=<%=bb.getPid()%>'>删除</a></td>
            </tr>
        <% index ++;
        } %>
            <tr>
                <td colspan="7" align="right">
                    重量总计：1.25kg   原始金额：￥
                    <%= pro.getTotal() %>
                    元 - 返现：￥0.00 元<br/>
                    商品总金额：￥ <%= pro.getTotal() %>元
                </td>
            </tr>
        </form>
        </table>
    </body>
</html>
```

希望你尽可能的自己实现，在这个过程中会遭遇到很多问题，解决这些问题的过程，便是从学习者成长为有经验的程序员的过程。

第 6 章

使用自定义标记 TAG

使用 JavaBean，我们已经将 JSP 文件中大量的 Java 语句分离到了纯粹的 Java 代码中了，JSP 文件的复杂度大幅度下降，使得很多 HTML 出身的程序员也能一同完成工作，但是我们发现单纯使用 JavaBean，无法彻底的消除 JSP 中的 Java 语句，仔细分析便知道那些无法分离出去的 Java 代码主要的作用就是显示 Java 中的变量，或者更简单地说是显示功能无法使用 JavaBean。

这样 JSP 引入了自定义标记，标记我们都很熟悉<html>就是一个标记，自定义标记就是在 HTML 标记以外，我们在程序中定义的有含义的标记。

现在我们先来写一个最简单自定义标记，体会一下如何实现，就用自定义标记在网页上输出“学习 java web”。

首先编写 Java 代码，在代码中 TAG 代表标记，所以我们的类就叫做 ShowTag。

```
package com.wy.tag;

import javax.servlet.jsp.tagext.*;

public class ShowTag extends TagSupport{
    public int doStartTag() {
        try {
            pageContext.getResponse().setContentType("text/html;charset
=gb2312");
            pageContext.getOut().println("学习 java web");
        }catch(Exception e){}
        return TagSupport.SKIP_PAGE;
    }
}
```

这段代码有些特别，首先你要扩展 JDK 的 Jar，扩展的方式和数据库、Servlet 一样，文件也在 Tomcat 目录下的 lib 目录中，之前扩展 Servlet 的时候，文件也在这里，Servlet 的 jar 文件叫做 servlet-api.jar，现在要扩展自定义标记的 jar 文件叫做 jsp-api.jar。

从代码中能够看出来，自定义标记的类从 TagSupport 继承而来，覆盖的方法是 doStartTag，或许你能够猜想出来应该还有一个 doEndTag，记得在 HTML 中标记通常是有开始、有结束的，所以这两个方法也对应着开始和结束的标记，等我们成功实现了这个最简单的标记后，你自己来试试结束标记。

特别的是 doStartTag 需要一个 int 型的返回值，现在我们来看 return 语句，如我所料，这里使用了静态的常量来满足 int 值的需要，用 TagSupport 点点会看到那些备选项，看英文的意思，大体能猜到每个选项的含义。这些选项总体上表达了两个方面的含义，首先是包含或是跳过，在 HTML 中，很多标记的开始和结束之间会包含内容，那么跳过就意味着会忽略这些包含的内容，或是说让标记包含的内容无效。第二层含义就是包含或跳过多大范围。具体你自行尝试。

我们现在的任务非常简单，就是输出，事实上自定义标记最主要的使命就是实现输出，我们知道输出依靠的是 response 对象或是 out 对象，但是有时为了输出我们不得不使用 session，要重针对 Tag 再次准备一堆内置对象似乎有点麻烦，其实 JSP 标准在定义的时候就考虑到了这个问题，事先就准备了一个内置对象 pageContext，直接看英文意思是页面的上下文，我们来理解页面的上文是 request，下文是 response，自身是 page、out，上文、自身和下文都能看到的是 session，pageContext 对象中包含了其他的所有内置对象。所以能看到我们用 pageContext.getOut()来获得 out 对象。

由于在输出的字符串中包含了中文，所以我们输出前要设置内容的类型，这和 Servlet、JSP 没有区别。

现在我们来思考一下在什么情况下，我们需要 TAG。一直我都在讲 JavaBean 和 JspTag 技术的初衷都是为了将 Java 代码从 JSP 中分离出去，一旦这个目的达到了，人们就会发现 JspTag 另外的价值。

人们发现自定义标记是 Web 开发过程中非常好用的代码重用技术，比如，作为一个 Web 程序员，我们会在很多项目中编写用户登录页面，假设写好这个页面需要 50 行代码，现在使用自定义标记，就可以用一个标记代替这 50 行代码，这样我们就可以全力以赴的写一个完美的用户登录，以后再遇到这样的任务，直接用自定义的标记就好了。

这就意味着，一个老的 Web 程序员可能手头积累了大量的标记，再开发新的系统时，有很多通用的内容都不需要从头再写了，开发效率从客户的角度来看，成倍的提升，这就不是新手能够做到的，如果一个软件企业有意识地这样积累自定义标记，这个软件企业的开发成本就会越来越低，开发效率就会越来越高，这些自定义标记就会成为企业的重要资产，所以人们常常会称这项技术为自定义标记库，那么如何管理这个拥有很多自定义标记的库呢？我们再次求助于 XML 文件，只不过新的扩展名是 tld，即便想我们现在只有一个标记，也要通过 tld 对标记进行管理。

打开软件 XMLSpy，在新建的对话框中，你能够找到创建 tld 的选项，我们会得到一个拥有基本选项的 tld 文件，现在添上内容。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE taglib PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library
1.2//EN"
    "http://java.sun.com/dtd/web-jsptaglibrary_1_2.dtd">
<taglib>
    <tlib-version>1.0</tlib-version>
    <jsp-version>1.1</jsp-version>
    <short-name>wytag</short-name>
```

```
    <tag>
        <name>show</name>
        <tag-class>com.wy.tag.ShowTag</tag-class>
    </tag>
</taglib>
```

版本信息只需要照着写就好了，事实上现在已经支持更高的版本了，只是现阶段我们还感觉不到不同版本之间的差别，short-name 的值我设定的是 wytag，这个文件存盘的时候就叫做 wytag.tld，其实这两个名字可以不同，但是我个人认为没必要为此多起一个名字。

事实上 tld 文件中，会有很多 tag 标记，这个很好理解，因为我们会有很多标记，只是现在我们只有一个标记，我们看到 ShowTag 这个自定义标记的类，对应的名字是 show，现在我来总结一下，到目前为止，我们遇到了这样几个名字，com.wy.tag.ShowTag 是类名，show 是标记的名字，wytag 是标记库描述文件的 short-name 和文件名。

最后，我们将这个文件保存到 Tomcat 中，位置和 web.xml 一致，毕竟这也是个 XML。

有没有想过，如果有一天你也成为老程序员，你手里积累了几百个自定义标记，几乎可以应对大多数 Web 开发的任务，有没有可能把这些标记卖给新的 Web 程序员，这样新的程序员也可以轻轻松松地快速开发了，换个角度看，如果有人卖这样的标记库，你是不是也会买一个。事实上确实有人这么做，其中买的最成功的标记库叫做 JSTL，这个缩写的意思是“JSP 标准标记库”，这个标记库不要钱，和 JSP 一同提供给你，既然这样，确实有必要学会 JSTL。

只是 JSTL 太复杂了，几乎是又开发了一门新的计算机语言，我向来不会使用罗列的方式讲授技术，但是讲授 JSTL 似乎没有多好的办法，从另一个方面来看，如果你能真正理解自定义标记，甚至能够自己来编写，学会 JSTL 是件太简单的事情，毕竟一个是开发，一个是使用。

现在我们来编写 JSP 文件，为了试验新建一个文件 test.jsp。

```
<%@ taglib uri="/WEB-INF/wytag.tld" prefix="wy"%>
<html>
    <body>
    <wy:show/>
    </body>
</html>
```

看到在第一句话，我们使用<%@ taglib%>将标记库描述文件引入，并且命名为 wy，还记得在<%@ %>这个语句形式中，我们都学习过什么吗？首先是 page，一些对页面整体进行设置的动作，再有，include 可以将另外一个页面包含进来，现在还有 taglib。<%@ %>语句形式只有这三个大类别。

<wy:show/>便是使用我们自定义的标记。现在启动 Tomcat，然后访问这个 test.jsp 就可以看到我们输出的那句话了。

我想你可以试着自己编写一个用户登录的标记，一定要尽可能的写得完美，或许以后工作就用它了。

6.1　使用 JSP、JavaBean 和 TAG 实现商品显示

这个代码在之前 JSP+JavaBean 的基础上添加，目的就是将用于显示的 Java 代码分离到 Tag 中。

找到并看一下之前的代码，我们需要写两个标记，一个用于显示商品，一个用于显示页码。

先来看显示商品的标记类文件，我将它命名为 ShowProduct，由于之前已经使用了 JavaBean，而且这个 JavaBean 的作用域是 session，所以在自定义标记的 Java 代码中，可以从 session 里找到这个 pro 的 JavaBean，这样我们就能够得到数据了。

还有个问题，我们在这个自定义标记中是提供整个表格的 HTML 代码，还是放过表头提供表格的内容。我认为应该放过表头，这样编写 HTML 的页面程序员就有机会按照自己的想法来美化这个表格了。

下面我们看 Java 代码，首先写好自定义标记库的代码框架，我们的第一个任务是到 session 中找到那个 JavaBean，还记得有个 pageContext 对象吗，它其中包含着所有的 JSP 内置对象，要从这个 pageContext 中找到 session 对象，这个 JavaBean 的名字就是使用 useBean 标记时的 id，在我们的代码中名字是“pro”。到目前为止的代码如下。

```
package com.wy.tag;

import javax.servlet.jsp.tagext.*;
import javax.servlet.http.*;
import com.wy.bean.Products;

public class ShowProduct extends TagSupport{
    public int doStartTag() {
        try {
            //获得 JavaBean
            HttpSession session = pageContext.getSession();

            Products pro = (Products)session.getAttribute("pro");
        }
        catch (Exception ex) { }
        return TagSupport.SKIP_BODY;
    }
}
```

现在可以调用 JavaBean 的方法来得到要显示的内容，并且按照 HTML 表格的语法格式输出。

```
package com.wy.tag;

import javax.servlet.jsp.tagext.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;
import com.wy.bean.Products;
import com.wy.bean.ProBean;
```

```
import java.util.*;

public class ShowProduct extends TagSupport{
    public int doStartTag() {
        try {
            //获得 JavaBean
            HttpSession session = pageContext.getSession();

            Products pro = (Products)session.getAttribute("pro");
            HashSet<ProBean> pbs = pro.getProducts();

            int cell = 0;//用于管理输出<tr></tr>
            JspWriter out = pageContext.getOut();
            for(ProBean pb : pbs) {
                //输出 HTML 内容
                if(cell%pro.getCells()==0) {
                    out.println("<tr align='center'>");
                }
                cell ++;
                out.println("<td>");
                //图片
                out.println("<img src='"+pb.getImg()+"'/>");
                //商品名称
                out.println("<div id='name'>");
                out.println("<a href='' title='");
                out.println(pb.getInfo());
                out.println("'> "+pb.getName()+"</a>");
                out.println("</div>");
                //显示单价
                out.println("<div class='price'>&yen;");
                out.println(pb.getPrice()+"</div>");

                //显示评论
                out.println("已有"+pb.getComment()+"人评论");

                //显示赠品图标
                if(pb.getGift() !=0){
                    out.println("<a class='p1' title='购买本商品送赠品'>");
                    out.println("<img src='img/gift.png'></a>");
                }

                //显示直降图标
                if(pb.getDown().equals("y")){
                    out.println("<a class='p2' title='本商品正在降价销售中
'>");
                    out.println("<img src='img/down.png'></a>");
                }
```

```
                //显示按钮
                out.println("<div>");
                out.print("<input  type='button'  value=' 购 买 'onclick=\
"location.replace('products.jsp? pid=");
                out.print(pb.getPid()+"');\">");
                out.print("<input type='button' value='关注'>");
                out.print("<input type='button' value='对比'>");
                out.print("</div>");
                out.print("</td>");
                if(cell%pro.getCells()==0) {
                    out.println("</tr>");
                }
            }
        }
        catch (Exception ex) {}
        return TagSupport.SKIP_BODY;
    }
}
```

这个代码实现起来并不容易，我们似乎又回到了 servlet 时代，单单那些单引号双引号就让人头大。

下面我们来实现显示页码的代码，这个简单多了。

```
package com.wy.tag;

import javax.servlet.jsp.tagext.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;
import com.wy.bean.Products;

public class ShowPage extends TagSupport{
    public int doStartTag() {
        try {
            //获得 JavaBean
            HttpSession session = pageContext.getSession();

            Products pro = (Products)session.getAttribute("pro");

            JspWriter out = pageContext.getOut();

            for(int i = 1; i <= pro.getPageCount(); i ++) {
                out.println("<a href='products.jsp?nowPage="+i+ "'>"+i+"
</a>");
            }
        }catch (Exception ex) {}
        return TagSupport.SKIP_BODY;
    }
}
```

两个代码都编译没有问题了，我们要改写 wytag.tld，标记库的描述文件，添加这两个标记。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE taglib PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag
Library 1.2//EN"
    "http://java.sun.com/dtd/web-jsptaglibrary_1_2.dtd">
<taglib>
    <tlib-version>1.0</tlib-version>
    <jsp-version>1.1</jsp-version>
    <short-name>wytag</short-name>
    <tag>
        <name>show</name>
        <tag-class>com.wy.tag.ShowTag</tag-class>
    </tag>
    <tag>
        <name>product</name>
        <tag-class>com.wy.tag.ShowProduct</tag-class>
    </tag>
    <tag>
        <name>page</name>
        <tag-class>com.wy.tag.ShowPage</tag-class>
    </tag>
</taglib>
```

有了上面的准备，具体的 jsp 文件就要简单太多了。

```
<%@ taglib uri="/WEB-INF/wytag.tld" prefix="wy" %>
<html>
<body>
<jsp:include page="menu.jsp"/>
<!-- 准备 -->
<%@ page contentType="text/html;charset=gb2312" %>
<%@ page import="java.util.*,com.wy.bean.ProBean" %>

<jsp:useBean id="pro" class="com.wy.bean.Products" scope="session"/>
<jsp:setProperty name="pro" property="*"/>
<!-- 显示当前记录 -->
<table width="100%">
<wy:product/>
</table>
<!-- 显示页码 -->
<wy:page/>
</body>
</html>
```

这段代码简单吧，大量烦琐的逻辑被隐藏到 JavaBean 中去了，而所有的显示代码则被封装到了 TAG 中。

你也可以将购物车的内容用 TAG 封装并提供给网页设计者，如果用 CSS 将购物车的内容定义成半透明的绝对定位，是不是能够轻松的显示在商品展示的页面中，每次选择购买一个新的商

品，页面的一侧就会实时的显示购物车中的商品，仔细做会有很酷的效果。

我们进一步的想法是，能不能让我们所编写的标记更加通用，这样就可以让我们的标记用到很多不同的项目中。比如显示商品，如果能够在使用标记的时候可以定义每行显示几个商品就会灵活多了。

自定义标记提供了一种可以使用参数的形式。我们先来看含有参数的自定义标记如何使用：<wy:product cells="2"/>，这样就出现了一个叫做 cells 的属性，那么如何将这个属性的值传递到类中呢？

想一下，在 Java 中我们曾经遇到过一个叫做属性的东西，在学习 JavaBean 的时候我们称 setter 和 getter 方法为属性，不知道是巧合，还是这个行业存在着默契，HTML 标记的属性和 JavaBean 中的属性是对应着的。

虽然在 JavaBean 中有属性提供了每行显示的商品数量，我又在自定义标记中提供设置每行显示的商品数量，这看起来有些多余，可是也提供给页面设计着更加丰富的选择，在具体使用标记的时候，这样就允许重新设置显示方式了。

```
package com.wy.tag;

import javax.servlet.jsp.tagext.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;
import com.wy.bean.Products;
import com.wy.bean.ProBean;
import java.util.*;

public class ShowProduct extends TagSupport{
    private int cells;
    public void setCells(int value) {
        cells = value;
    }
    public int doStartTag() {
        try {
            //获得 JavaBean
            HttpSession session = pageContext.getSession();

            Products pro = (Products)session.getAttribute("pro");
            HashSet<ProBean> pbs = pro.getProducts();

            int cell = 0;//用于管理输出<tr></tr>
            JspWriter out = pageContext.getOut();
            for(ProBean pb : pbs) {
                //输出 HTML 内容
                if(cell%pro.getCells()==0) {
                    out.println("<tr align='center'>");
                }
                cell ++;
```

```
                out.println("<td>");
                // 这部分代码省略，请参照上边的代码
                out.print("</td>");
                if(cell%cells==0) {
                    out.println("</tr>");
                }
            }
        }
        catch (Exception ex) {
        }
        return TagSupport.SKIP_BODY;
    }
}
```

现在我们要改写 tld 文件，以便 Tomcat 清楚有属性这回事。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE taglib PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag
Library 1.2//EN"
    "http://java.sun.com/dtd/web-jsptaglibrary_1_2.dtd">
<taglib>
    <tlib-version>1.0</tlib-version>
    <jsp-version>1.1</jsp-version>
    <short-name>wytag</short-name>
    <tag>
        <name>show</name>
        <tag-class>com.wy.tag.ShowTag</tag-class>
    </tag>
    <tag>
        <name>product</name>
        <tag-class>com.wy.tag.ShowProduct</tag-class>
        <attribute>
            <name>cells</name>
            <required>true</required>
            <rtexprvalue>false</rtexprvalue>
        </attribute>
    </tag>
    <tag>
        <name>page</name>
        <tag-class>com.wy.tag.ShowPage</tag-class>
    </tag>
</taglib>
```

在 tag 元素下面增加了属性 attribute 元素的描述，如果有多个属性就要并列多个 attribute，名字就是属性的名字，这里和 Java 代码中的属性以及自定义标记使用时的属性是一致的。

下面两个元素不是必须的，因为它们有默认值，required 说明这个属性是否是必须的，我们在使用 HTML 标记的时候发现，大多数属性可以没有，因为有默认值存在，在这种情况下，我们的 Java 代码中，cells 变量就需要有默认值。你也可以要求在使用标记的时候，必须提供某个

属性，否则会在访问的时候出错，在我的代码中，要求必须提供这个属性。

rtexprvalue 元素说明该属性是否可以通过表达式提供，通常我们会直接提供属性的值，比如写成这个样子<wy.:product cells="2"/>，但是有时我们可能会这样提供这个值<wy:product cells="<%= i%>"/>，这样的值就是在运行过程中，通过表达式提供的，这涉及到 Tomcat 要清楚是不是先运行里面的表达式，再调用自定义标记的代码，false 是不允许通过表达式提供，这种情况比较常见。

好了，现在我们改写 JSP 文件，体验一下加上参数的自定义标记的代码吧。

```
<%@ taglib uri="/WEB-INF/wytag.tld" prefix="wy" %>
<html>
<body>
<jsp:include page="menu.jsp"/>
<!-- 准备 -->
<%@ page contentType="text/html;charset=gb2312" %>
<%@ page import="java.util.*,com.wy.bean.ProBean" %>

<jsp:useBean id="pro" class="com.wy.bean.Products" scope="session"/>
<jsp:setProperty name="pro" property="*"/>
<!-- 显示当前记录 -->
<table width="100%">
<wy:product cells="2"/>
</table>
<!-- 显示页码 -->
<wy:page/>
</body>
</html>
```

虽然访问的结果会比较混乱，但是每页确实显示了两个商品。

到目前为止我们已经在 Java Web 的范畴里，拥有了第二层的能力，不仅仅能够实现动态网页，而且通过分离可以让程序相对模块化的实现，这样一个团队中不同的人，就可以各有所长的完成不同的技术工作，这些模块也更容易测试和重用。

在进入第三层之前，你需要通过大量的练习彻底掌握到目前为止我带领你体验过的内容，现在停下这本书，全力以赴的实现一个尽可能完整的 360buy.com，不仅仅是前台的页面，还要包含后台的管理系统，看起来需要事先规划一下整个工作，认认真真地设计好数据库的表结构。如果可以找到几个人形成一组最好不过了，这样可以体会到团队的协作。

Broadview®
www.broadview.com.cn
博文视点·IT出版旗舰品牌
技术凝聚实力·专业创新出版

Java Web开发就该这样学

学习本书的步骤

第一步：扔书。我会请所有的学生将学校发的书从窗户扔出去，看一百遍这些愚蠢的书，你也不可能学会这项技术。这本书也教不会你，真正能让你学习并一直坚持学下去的是最开始的兴趣，是整个学习过程持续不断的热情。

第二步：编程不用学理论。本书会给学习者一直带来成就感，即便没有基础，你也只需要2个小时便能够写出可以出去炫耀的“王八”。

第三步：你一直需要动手操作。没有人告诉你真正的答案，完全要你自己发现；内容被设计得既不容易也不难。正确的学习体验是，我提示你，你尝试，经过失败、反思和努力，你找到解决办法，并体会其中的感觉，直到你会了，或许你没法将你会的东西告诉别人，但是你会的已经是你自己的能力。

第四步：编程的学习曲线和代码规模有关系。主动练习的量、基础、逻辑能力等因素直接影响着本书学习的时间进程。

第五步：所有代码都要敲20遍。对编程的理解程度和他敲过的代码遍数成正比。

学习本书的建议

1. 在作者提问的地方，停顿一下，思考一下，再向后看。
2. 处理开始的代码，在其他代码前，作者都会描述是做什么的，请看完描述，不要直接看或练习本书提供的代码，而是放下书，尝试着自己实现，即便是失败了，再看我的代码，你也是主动学习，而不是被动接受。
3. 本书会在需要的地方提示你练习，或者建议你停下来明天再学，这些都是作者长期教学经验的积累，是学习曲线所提供的节奏，希望你能遵从这些建议。

本书配套源代码下载链接：www.broadview.com.cn/20453

上架建议：网站开发 > Java Web

ISBN 978-7-121-20453-1

9 787121 204531 >

定价：49.00元

策划编辑：孙学瑛
责任编辑：徐津平
封面设计：侯士卿